



---

# Konvojstyring af Mobile Robotter

Bachelorprojekt Rapport  
(Convoy Control of Mobile Robots)

---

Markus Kenno Hansen (s224044)

Supervisors: Søren Hansen & Nils Axel Andersen  
Danmarks Tekniske Universitet  
8. juni 2025

## 1 Abstract

This project investigates a LiDAR-based convoy system for DTU's Small Mobile Robots. A leader robot follows a pre-programmed path, while the follower robot uses distance and angle information from a LiDAR-sensor to maintain formation. A PI-Lead controller regulates the follower's velocity based on its distance to the robot ahead. Experimental results demonstrate that the system successfully maintains formation in the tested scenarios. Although some minor deviations and corner-cutting behaviors are observed while turning, the follower robots successfully end up in a line behind the leader when it completes its path.

# Indhold

<b>1 Abstract</b>	<b>1</b>
<b>2 Introduktion</b>	<b>4</b>
<b>3 Kinematisk Model</b>	<b>6</b>
3.1 Udledning af Hjulhastigheder . . . . .	6
3.2 Udledning af Retning . . . . .	7
3.3 Koordinattransformasjon . . . . .	7
<b>4 Hastighedsregulator</b>	<b>9</b>
4.1 Overføringsfunktion . . . . .	9
4.2 Steprespons . . . . .	9
4.2.1 Steprespons af SMR-Robot . . . . .	10
4.3 Regulator . . . . .	12
4.4 P-Regulator . . . . .	13
4.5 PI-Regulator . . . . .	14
4.6 PI-Lead-Regulator . . . . .	15
4.7 Diskretisering af Regulator . . . . .	16
<b>5 Centroid-Detektion via LiDAR data</b>	<b>19</b>
5.1 Clustering . . . . .	20
5.2 Centroid . . . . .	22
<b>6 Implementering af Hastighedsregulator &amp; Centroid-detektion</b>	<b>24</b>
6.1 Justering af hastighedsregulator . . . . .	24
6.1.1 Steprespons af hastighedsregulator . . . . .	29
6.1.2 Steady-State Error . . . . .	30
6.2 Justering af distance regulering . . . . .	32
6.2.1 Hjørneskæring . . . . .	33
6.2.2 Centroid projekton . . . . .	33
6.3 Søgeområde . . . . .	34
6.4 Fallback . . . . .	35
6.5 Endelig implementering . . . . .	35
<b>7 Test af Implementering</b>	<b>38</b>
7.1 Resultater . . . . .	40
7.1.1 Odometri . . . . .	40
7.1.2 Video Analyse . . . . .	42
<b>8 Resultatbehandling</b>	<b>45</b>
8.1 Rå odometri-data . . . . .	45
8.2 Analyse af Odometri-data . . . . .	46
8.2.1 Analyse af Afgangelse for Ruterne . . . . .	50

8.3	Video Analyse . . . . .	52
8.3.1	Korrektion af Filmvinkel . . . . .	52
8.3.2	Afgivelse medhensyn til Førerrobotten . . . . .	55
8.3.3	Afgivelse medhensyn til Forankørende . . . . .	57
<b>9</b>	<b>Diskussion</b>	<b>59</b>
9.1	Cluster-densitet . . . . .	59
9.2	Centroid (Odometri) . . . . .	59
9.3	Forankørende (Videodata) . . . . .	62
9.4	Førerrobot (Videodata) . . . . .	63
9.5	Kvalitativ Analyse . . . . .	63
9.6	Forbedringsforslag og Videreudvikling . . . . .	64
<b>10</b>	<b>Konklusion</b>	<b>65</b>
<b>11</b>	<b>Bilag</b>	<b>69</b>

## 2 Introduktion

Danmarks Tekniske Universitet har en række robotter, der er blevet anvendt til forskellige formål, blandt andet undervisning. En af disse robotter er de såkaldte Small Mobil Robotter (herefter kaldes **SMR** robotter), som ses på figur 1. Robotterne er kategoriseret som 'Differential drive robots', hvilket betyder at de er udstyret med to uafhængige DC-motorer, som driver baghjulene. Til at navigere er de blandt andet udstyret med en Hokuyo URG-04LX LiDAR (laser image, detection, and ranging), samt et Xbox-360 Kinect kamera med mere. Alle komponenter er forbundet og styret af en NUC-computer. Til dette projekt vil LIDAR-sensoren, computeren og DC-motorerne udlukkende bruges til at implementere et system, som tillader robotterne at køre i en konvoj. Systemet vil implementeres i programmeringssproget C, og 'Small Mobile Robot Control Language' (**SMR-CL**)[[AR04](#)], hvor **C-programmet** sender de nødvendige **SMR-CL** kommandoer som robotten eksekverer.



Figur 1: Billede af de fire **SMR** robotter, som indgår i konvojen, i deres startposition.

Formålet med rapporten er at anvende relevant teori til at udlede et **C-program**, som anvender LiDAR-sensoren til at identificere en forankørende robot og følge samme rute som denne. For at realisere dette implementeres en hastighedsregulator i **C-koden**, som udregner de hastigheder som hvert hjul skal køre med for at følge den forankørende robot. En kvantitativ og kvalitativ analyse af hvor godt robotterne følger efter hinanden i konvojen udarbejdes. I analysen bliver både robotternes indbyggede odometri-data analyseret, samt en uafhængig videooptagelse af ruten for at opnå et uafhængigt datasæt af deres præstationer.

Rapporten gennemgår slavisk designprocessen af programmet, samt gennemgås det relevante teori undervejs. I Afsnit 3 udledes de matematisk funktioner for hjulhastighederne,

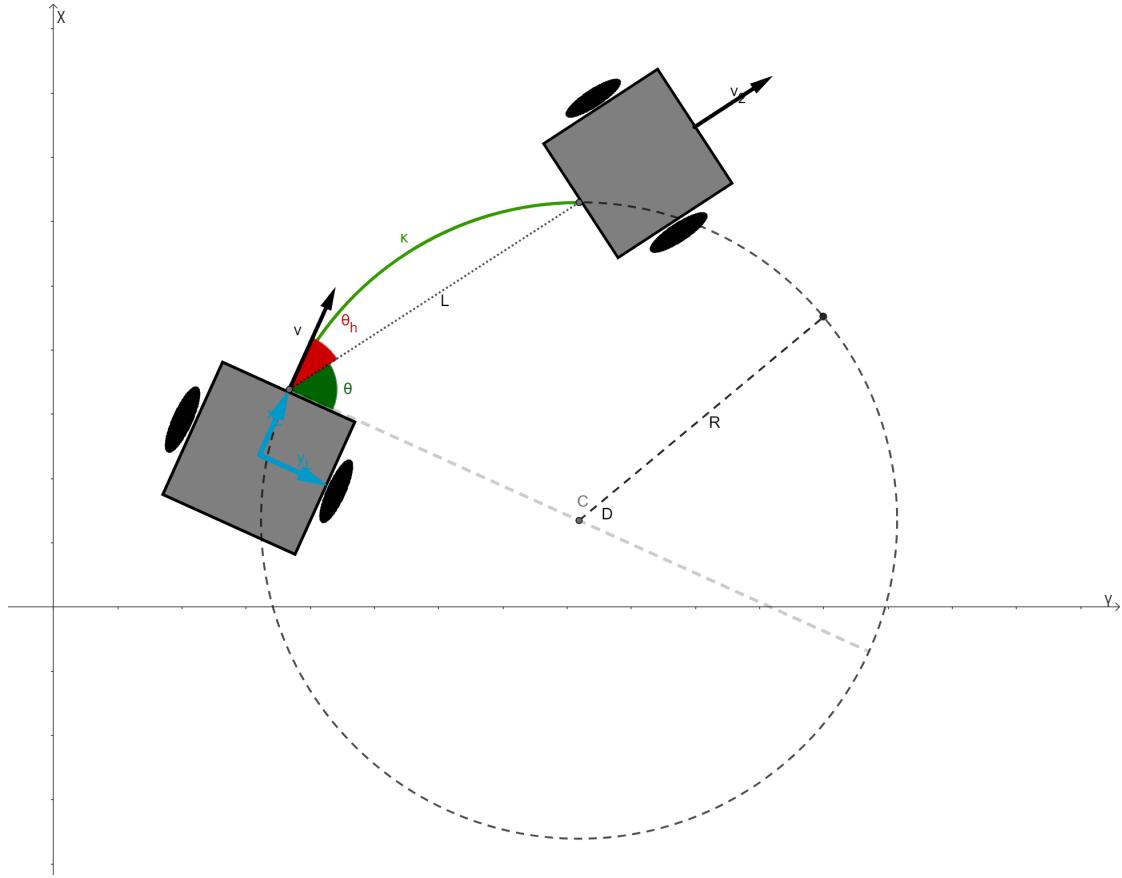
## 2 INTRODUKTION

---

som så anvendes som hastighedskommandoer, hvor der i afsnit 4 bliver gennemgået designprocessen af regulatoren til hastighederne. Afsnit 5 omhandler realtid databehandlingen af LiDAR-sensorens data, og hvordan en algoritme anvendes til at styre robotterne. Afsnit 6 beskriver hvordan hastighedsregulatoren og algoritmen implementeres, og i afsnit 7 beskrives forsøgsopstillingen, som skal afprøve denne implementering. Resultaterne bliver analyseret og diskuteret i henholdsvis afsnit 8 og 9, som danner basis for konklusionen i afsnit 10.

### 3 Kinematisk Model

Da programmet styrer robotterne ved at give specifikke hjulhastighedskommandoer, er det essentielt at forstå, hvordan kommandoerne til hvert hjul udregnes. For at udlede de relevante ligninger, opstilles en kinematisk model, som bruges til at udlede de individuelle hjulhastigheder. Derudover udledes en basisskiftematrice, som anvendes til at konvertere robotternes individuelle lokale koordinatsystemer til ét fælles globalt koordinatsystem.



Figur 2: Skitse af konvojsystem med relevante variabler for opstilling af kinematiske modeller.

#### 3.1 Udledning af Hjulhastigheder

Første opskrives en kinematisk model for systemet. Det gælder, at for hver robot kan deres hastighedsvektor  $\mathbf{v}$  skrives som

$$\mathbf{v} = v(t) \cdot \begin{bmatrix} \cos(\theta_h) \\ \sin(\theta_h) \end{bmatrix} = \begin{bmatrix} \frac{d}{dt}x(t) \\ \frac{d}{dt}y(t) \end{bmatrix}$$

hvor  $v$  er skalar-hastigheden for robotten, og  $\theta$  er den kurs / retning (heading), som robotten har. Hastigheden  $v$  er afhængig af begge hjulhastighederne således

$$v(t) = \frac{1}{2} \cdot (v_R(t) + v_L(t)) \quad (3.1)$$

hvor  $v_R(t)$ ,  $v_L(t)$  er henholdsvis højre-og venstre hjuls lineære hastighed. Hvor hurtigt kurven ændre sig i forhold til tid  $\frac{d}{dt}\theta(t) = \omega(t)$  kan ligeledes udtrykkes ved

$$\frac{d}{dt}\theta(t) = \omega(t) = \frac{v_R(t) - v_L(t)}{b} \quad (3.2)$$

hvor  $b$  er distancen mellem hjulene (basen). Løses ligningssystemet (3.1) og (3.2) for  $v_R(t)$  og  $v_L(t)$  udledes følgende udtryk for hvert hjuls hastighed.

$$\begin{aligned} v_R(t) &= v(t) \cdot \left(1 + \frac{b}{2} \cdot \omega(t)\right) \\ v_L(t) &= v(t) \cdot \left(1 - \frac{b}{2} \cdot \omega(t)\right) \end{aligned} \quad (3.3)$$

### 3.2 Udledning af Retning

Relationen mellem vinkelhastighed og lineær hastighed er defineret som  $\omega(t) = \frac{v(t)}{R}$ , hvor  $R$  er radius for cirkel rotationen, hvoraf det er gældende at  $\frac{1}{R} = \kappa$  er krumming af cirklen [Web25]. Yderligere fra basal cirkelregning udledes  $R = \frac{2\sin(\theta)}{L}$ , hvor  $L$  er kordens længden, [Reg25], mere specifikt for analysen, så er  $L$  afstanden til den forankørende robot, og  $\theta$  er vinklen til den samme robot. Da formålet er at få robotterne til at følge efter hinanden, da gælder  $\theta_{err} = \theta - \theta_h$ , hvor  $\theta_{err}$  er fejlen i den ønskede retning, hvilket giver det endelige udtryk for krumningen

$$\kappa = \frac{1}{R} = \frac{2\sin(\theta_{err})}{L} \quad (3.4)$$

Indsættes alt dette i ligning (3.3) opnåens det endelige udtryk for begge hjulhastighederne:

$$\underline{\underline{v_R(t) = v(t) \cdot \left(1 + \frac{\kappa \cdot b}{2}\right), \quad v_L(t) = v(t) \cdot \left(1 - \frac{\kappa \cdot b}{2}\right)}}$$

### 3.3 Koordinattransformation

Hver robot har sit eget lokale koordinatsystem, som skal transformeres til det globale koordinatsystem. Den bagerste robots startposition anvendes som det globale  $(x_g, y_g) = (0, 0)$ . Herfra kan alle andre robotters globale position udregnes via rotationsmatricen  $\mathbf{R}(\theta)$ :

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \quad (3.6)$$

Det endelige koordinatskifte gives ved

$$\begin{bmatrix} x_g \\ y_g \end{bmatrix} = \begin{bmatrix} x_{\text{robot}} \\ y_{\text{robot}} \end{bmatrix} + \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\mathbf{R}(\theta)} \cdot \begin{bmatrix} x_{\text{local}} \\ y_{\text{local}} \end{bmatrix} \quad (3.7)$$

Dette er en vigtig og uundværlig del af afsnit 8 resultatbehandlingen. Det er åbenlyst, at regulering og kontrol af hastigheden er essentielt, da uden kontrol vil det være umuligt at holde en ansvarlig afstand til den forankørende robot.

## 4 Hastighedsregulator

Formålet med hastighedsregulatoren er at få en robot til at opnå den ønskede hastighed uden at overskride den givne hastighedsværdi. Eksempelvis hvis en hastighedskommando på  $0.4m/s$  er givet, så skal robotten opnå denne hastighed, og ikke  $0.5m/s$ . For at udlede en hastighedsregulator skal robottens overføringsfunktion først identificeres.

### 4.1 Overføringsfunktion

For et kontinuert, lineært, og tidsafhængigt system gælder, at overføringsfunktionen er defineret ved:

$$G(s) = \frac{Y(s)}{X(s)} = \frac{\mathcal{L}\{y(t)\}}{\mathcal{L}\{x(t)\}} \quad (4.1)$$

hvor  $s \in \mathbb{C}$ , således at  $s = \sigma + j\omega$ , hvor  $j$  er den imaginære enhed, og  $\mathcal{L}$  er Laplace transformationen [ASN22]. Når en overføringsfunktion er fundet vil den ofte simplificeres til formen

$$G(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0}, \quad a_i, b_i \in \mathbb{R}$$

hvor de  $s$ -værdier, der henholdsvis får tælleren og nævneren til at gå mod 0 kaldes for 'zeros' (nulsteder) og 'poles' (poler). Dette kan ofte ses mest tydeligt på overføringsfunktions faktoriseret form:

$$G(s) = \frac{(s - z_1)(s - z_2) \cdots (s - z_m)}{(s - p_1)(s - p_2) \cdots (s - p_n)} \quad (4.2)$$

hvor  $z$  er zeros, og  $p$  er poles. Placeringen af polerne afgør udlukkende, om et system er stabilt. Et stabilt system vil for alle  $p$  opfylde  $Re\{p\} < 0$ , altså at alle polerne er mindre end 0, og dermed befinner sig i den venstre halvdel af det komplekse-plan (LHS). Hvis bare én pol befinner sig på den højre halvplan (RHS)  $Re\{p\} > 0$  er system ustabilt. I tilfældet  $Re\{p\} = 0$  kaldes systemet for marginalt stabilt. Det er selvsigende, at stabilitet er altafgørende for, om en hastighedsregulatoren kan bruges i praksis.

Stabilitet og overføringsfunktioner er tæt sammenknyttet med den såkaldte steprespons. De er så sammenknyttet, at hvis enten en overføringsfunktionen eller en steprespons er kendt, kan den ene udledes ud fra den anden.

### 4.2 Steprespons

En steprespons af et system opnås ved at påtrykke systemet med en stepfunktion, også kaldt en Heaviside-funktion  $u(t)$ , som er defineret som

$$u(t) = \begin{cases} 0, & t \leq 0 \\ 1, & t > 0 \end{cases}$$

$$U(s) = \mathcal{L}\{u(t)\} = \frac{1}{s}, \quad s \neq 0$$

Det vil sige, at for en arbitrer ordinær lineær differentialligning (ODE) gælder:

$$a_n y^{(n)}(t) + a_{n-1} y^{(n-1)}(t) + \cdots + a_0 y(t) = b_m u^{(m)}(t) + \cdots + b_0 u(t)$$

da for  $t < 0$  er  $u(t) = 0$ , som bare er den homogene differentialligning, og for  $t > 0$  er  $u(t) = 1$ .

Ved at anvende Laplace transformationen på begge sider opnås

$$\begin{aligned}\mathcal{L} \left\{ y^{(k)}(t) \right\} &= s^k Y(s) \\ \mathcal{L} \{ u(t) \} &= \mathcal{L} \{ 1 \} = \frac{1}{s}\end{aligned}$$

når anvendt på ODE'en giver (ved hjælp af lineariteten af Laplace transformationen)

$$(a_n s^n + a_{n-1} s^{n-1} + \cdots + a_0) Y(s) = (b_m s^m + \cdots + b_0) \cdot \frac{1}{s}$$

Herefter isoleres  $Y(s)$ :

$$Y(s) = \frac{b_m s^m + \cdots + b_0}{s(a_n s^n + \cdots + a_0)} = \frac{G(s)}{s}$$

Ved at benytte den inverse-Laplacetransformation på  $Y(s)$  udledes  $y(t)$

$$\mathcal{L}^{-1} \{ Y(s) \} = y(t), \quad (Q.E.D.).$$

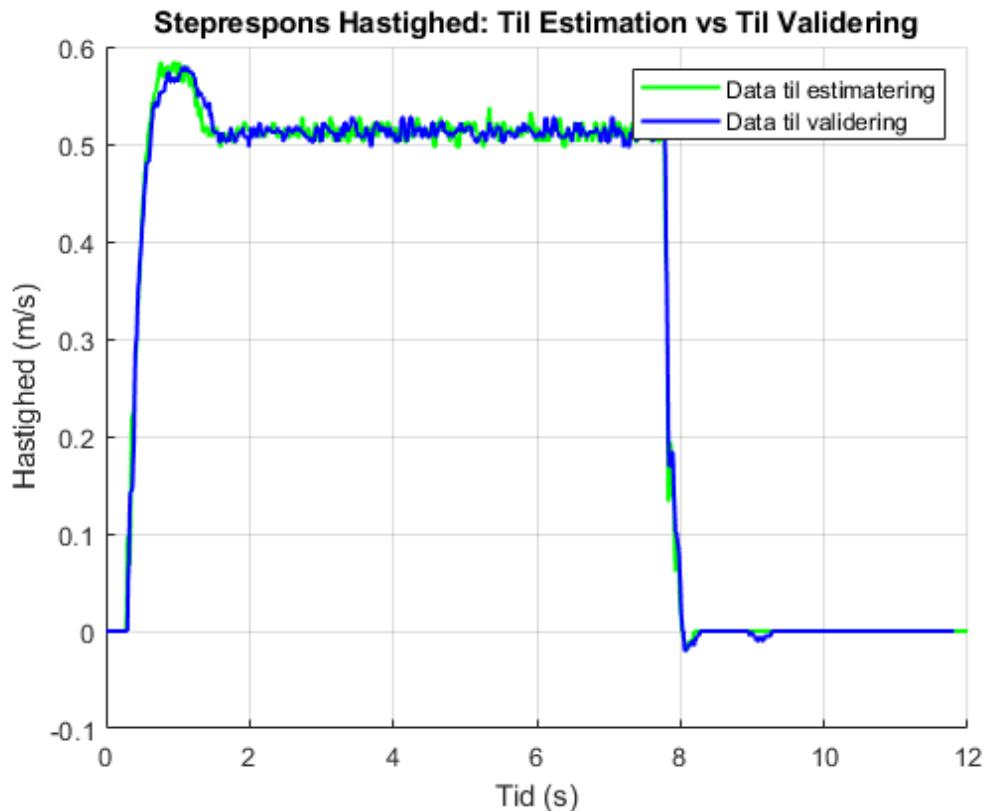
Deraf er det bevist, at stepresponsen af et system og systemts overføringsfunktion er sammenknyttet. Dette medfører, at en overføringsfunktion for **SMR** robotterne kan estimeres.

#### 4.2.1 Steprespons af **SMR**-Robot

Ved at sende en hastighedskommando til en **SMR**-robot, og anvende den indbyggede odometri til at logge dataen, blev en steprespons af **SMR**-robotten fundet (vist i figur 3). Med andre ord, blev **SMR**-Robotten påtrykket med en skaleret steprespons.  $0.5m/s^1$

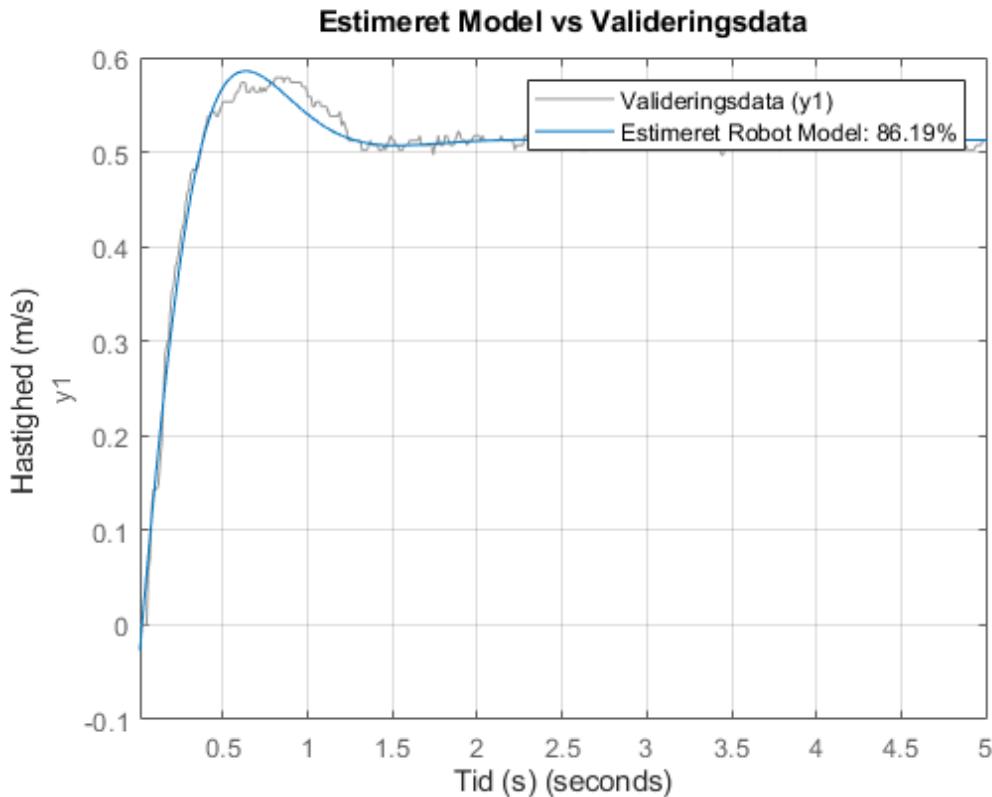
---

<sup>1</sup> $0.5m/s$  blev valgt for at undgå hjul skred.



Figur 3: To datasæt, der viser stepresponsen af SMR-Robot ved en hastighedskommando på  $0.5 \frac{m}{s}$ .

Der observeres et lille overshoot, samt lidt støj (som er uundgåeligt), og en ret hurtig setteling-time. Udfra dataen blev MATLABs `tfest` funktion brugt til at estimere en overføringsfunktion for systemet. Resultatet ses i figur 4.



Figur 4: Overføringsfunktion sammenlignet med valideringsdata med en NRMSE (Normalized Root Mean Square Error) fitness-værdi på 86.19% [Mat25].

Dette giver systemets overføringsfunktion

$$G(s) = \frac{21.1}{s^2 + 5.705s + 20.58} \quad (4.3)$$

med polerne  $p = -2.85 \pm j3.53$ , hvilket medfører systemet et stabilt. Udfra dette kan en regulator nu opskrives, hvis formål er at justere og kontrollere hastigheden.

### 4.3 Regulator

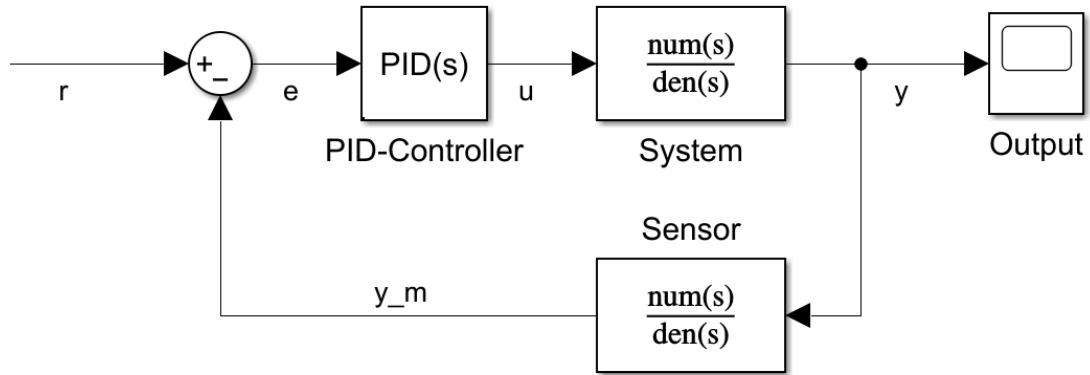
Formålet med en regulator er at have et system  $G(s)$ , tage dets output  $y(s)$  og føre det tilbage i systemet (feedback) efter det er blevet behandlet af en regulator. Ofte ønskes det at regulere efter en reference  $r$ . Forskellen mellem referencen og output kaldes for fejlen (error)  $e$ . Regulatorernes mål er at give et signal ( $u(s)$  eller  $u(t)$ ) an på domænet) til systemet, således at systemet giver den ønskede respons.

Det er således, at der skelnes mellem den åbne sløjfe (open-loop) og den lukkede sløjfe

(closed-loop) for et reguleret system.

$$\text{Open loop: } G_{ol}(s) = C(s) \cdot G(s)$$

$$\text{Closed loop: } G_{cl}(s) = \frac{G_{ol}(s)}{1 + G_{ol}(s)H(s)} = \frac{C(s)G(s)}{1 + C(s)G(s)H(s)}$$



Figur 5: Et reguleringssystem med feedback og relevante signalnavne.

Hvis der eksempelvis er en sensor  $H(s)$  i feedback loopet, så skal denne inkluderes i nævneren. En af de mest anvendte regulatortyper er en PID-Controller, og det er denne type, der vil blive anvendt i dette projekt. For at vurdere om en regulator er god, ses der på Steady-State Error  $e_{ss}$  defineret ved

$$\lim_{t \rightarrow \infty} e(t) = \lim_{s \rightarrow 0} sE(s)$$

Steady-State Error beskriver, afvigelsen mellem et reference input (i dette tilfælde en steprespons) og den endelige værdi som systemet ender med konvergente til.

#### 4.4 P-Regulator

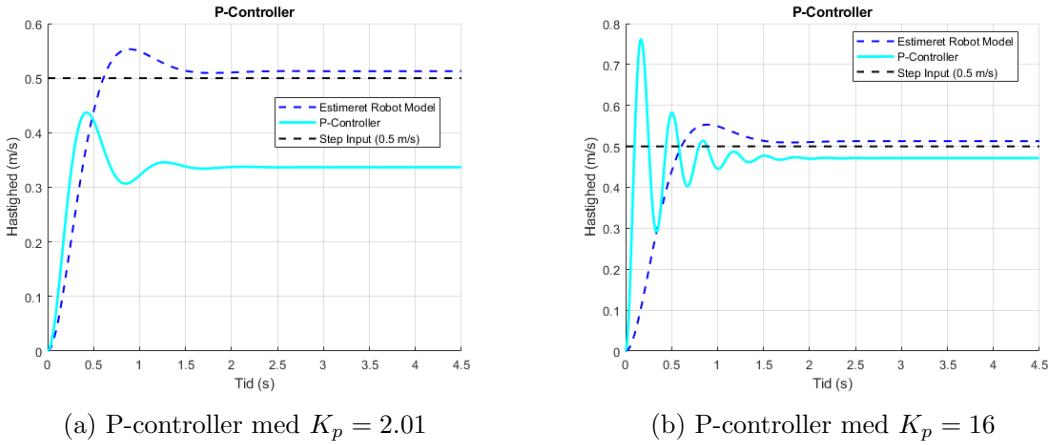
Den simpleste form for regulator er en P-regulator (proportional controller), hvor idéen er at lave reguleringen proportional med fejlen  $e(s)$ , ved at gange en gain  $K_P$  på systemets input og derefter lukke feedback loopet. I henholdsvis tids- og frekvensdomænet beskrives fejlen ved

$$\begin{aligned} u(t) &= K_P \cdot e(t) \\ u(s) &= K_p \cdot e(s) \end{aligned}$$

hvor  $u$  er signalet produceret af regulatoren.

Når man designet en P-controller, er man ofte interesseret i at have en 'phase margin'  $\gamma_m = 60^\circ$  [ASN22, p. 211], og løse for den 'cross-over frequency'  $\omega_c$  (frekvensen hvor

gainen er  $0dB$ ), som opfylder det kriterie. Ved at rykke  $\omega_c$  ændres hvor hurtigt regulatoren reagerer, samt stabiliteten. To eksempler på stabile P-kontrollerer ses på figur 6 med forskellige 'proportional-gains', henholdsvis  $K_p = 2.01$  og  $K_p = 16$ .



Figur 6: Steprespons plots af P-controller med to forskellige gain-værdier.

Det ses at en P-controller, ikke alene er nok til at regulere systemet. Da en lille gain ikke opnår den ønskede værdi, og for en stor gain er der et signifikant overshoot. Begge har en Steady-State Error på henholdsvis 0.16 og 0.028. Der skal tilføjes noget til regulatoren for at kompensere for Steady-State Error'en og overshootet.

## 4.5 PI-Regulator

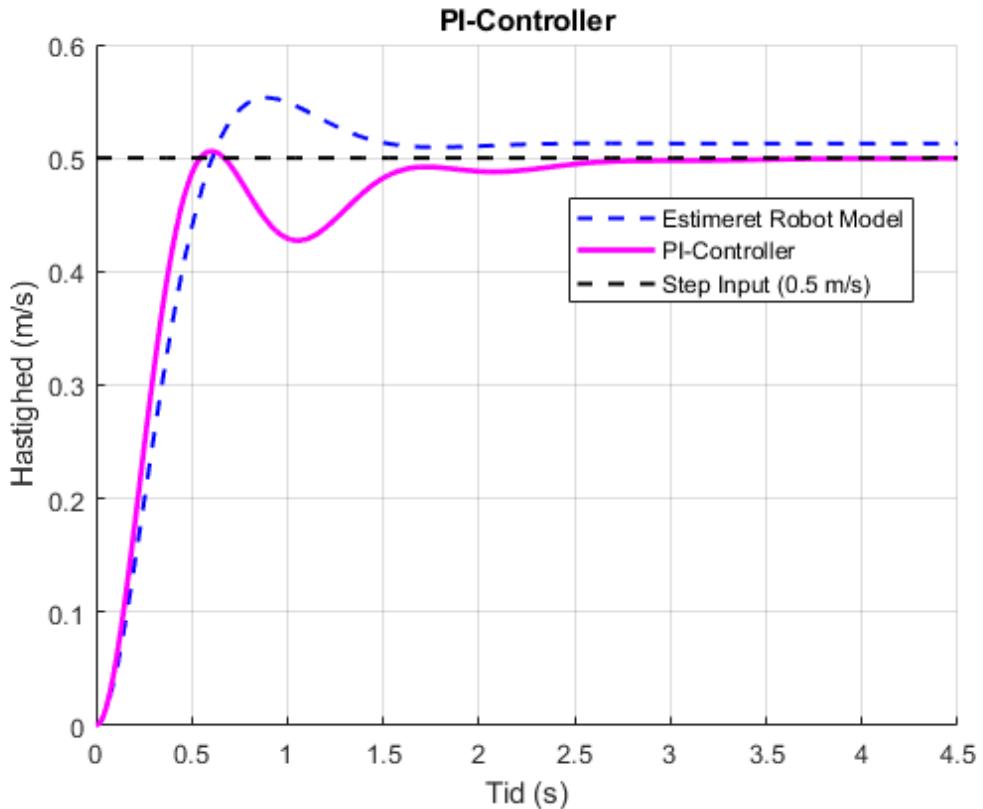
En PI-Regulator (Proportional Integral Controller) er en p-controller hvor der tilføjes et integral-term til regulatoren

$$\begin{aligned} u(t) &= K_P \left( e(t) + \frac{1}{\tau_i} \int_{\tau=0}^t e(\tau) d\tau \right) \\ u(s) &= K_P \cdot \left( 1 + \frac{1}{\tau_i s} \right) e(s) \end{aligned}$$

hvor  $\tau_i = \frac{N_i}{\omega_c}$ .  $\tau_i$  er integralets tidskonstant, og bestemmer hvor aggressivt den reagerer på akkumulerede fejl, mens  $N_i$  er 'integration tuning konstanten', som vælges ud fra hvad  $\omega_c$  er. Integral-termet akkumulerer dermed fejlene fra tidligere input over tid, og samtidig eliminerer små fejl. Den kompenserer dermed for State-State Error'en idet

$$\begin{aligned} e_{ss} &= \lim_{s \rightarrow 0} \frac{1}{1 + C(s)G(s)} \\ &= \lim_{s \rightarrow 0} \frac{1}{1 + K_P \left( 1 + \frac{1}{\tau_i s} \right) \cdot G(s)} \\ &= 0 \end{aligned}$$

I dette tilfælde vælges  $N_i = 2$  og  $\gamma_m = 60^\circ$ , for PI-controlleren, hvis steprespons er illustreret på figur 7 med en Steady-State Error  $e_{ss} = 0$ .



Figur 7: Stepsrespons for PI-Regulator.

Selvom det er en væsentlig forbedring, så er responsen stadig langsom, og der er nogle oscillationer.

## 4.6 PI-Lead-Regulator

I stedet for at tilføje et differential-term tilføjes i stedet et såkaldt 'Lead-term',

$$\frac{\tau_d s + 1}{\tau_d \alpha s + 1}, \quad \alpha \in ]0; 1[$$

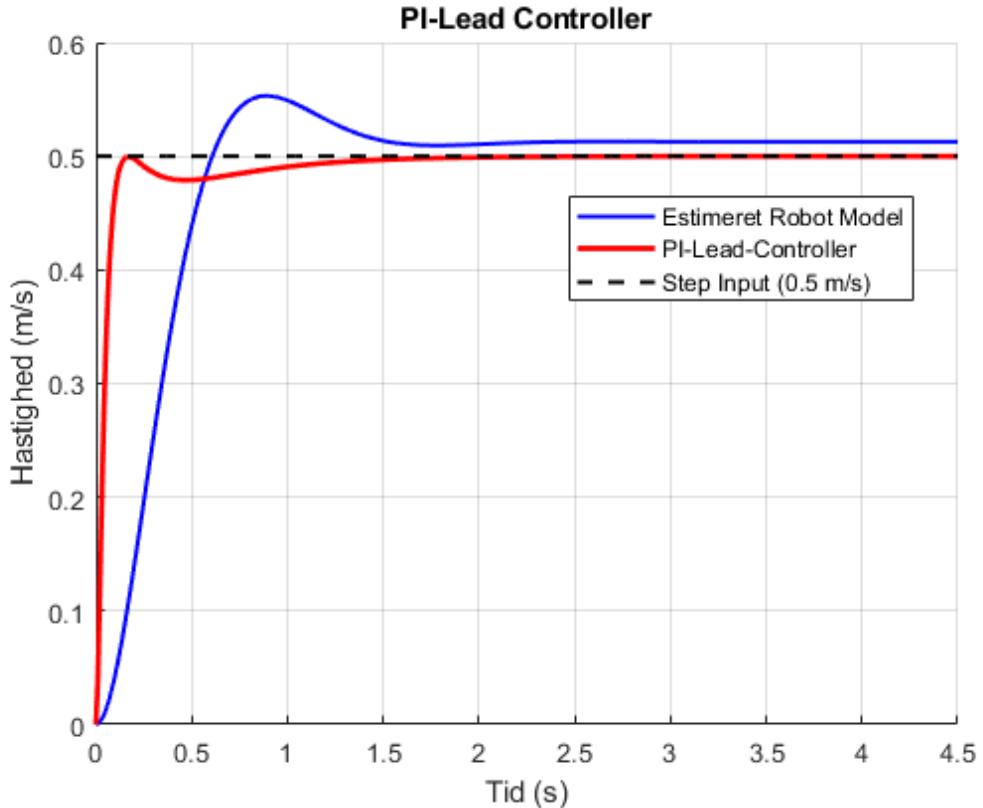
Matematisk kommer hele udtrykket til henholdsvis at være

$$u(t) = K_P \left( e(t) + \frac{1}{\tau_i} \int_0^t e(\tau) d\tau \right) + K_D \frac{d}{dt} \left( \frac{1}{1 + \alpha \tau_d \frac{d}{dt}} e(t) \right)$$

$$u(s) = K_P \left( 1 + \frac{1}{\tau_i s} \right) \cdot \left( \frac{1 + \tau_d s}{1 + \alpha \tau_d s} \right) e(s)$$

i tids- og Laplace domænet, hvor  $\tau_d$  er 'lead term konstanten', der bestemmer frekvensen af nulpunktet, og  $\alpha$  er 'lead factor', som bestemmer forholdet mellem polen og nulpunktet.  $\tau_d$  kan også udtrykkes som  $\tau_d = \frac{1}{\omega_c\sqrt{\alpha}}$ . Ved at optimere de forskellige tuningsparametre for den teoretiske bedst mulige respons for reguleringssystemet (hurtig responstid, hurtig setteling-time og lavt overshoot) udledes henholdsvis værdierne  $\alpha = 0.04$ ,  $N_i = 7$ ,  $\gamma_m = 75$ , hvilket resulterer i stepresponsen set i figur 8, hvor regulatorenens overføringsfunktion er

$$C_{ol}(s) = \frac{0.3255 s^2 + 2.36 s + 4.157}{0.003133 s^2 + 0.3311 s} \quad (4.4)$$



Figur 8: Step af respons af teoretisk optimal PI-Lead-Regulator.

Selvfølgelig, da SMR-rotterne bliver styret af diskrete hastighedskommandoer skal regulatoren først diskritiseres før den anvendes.

## 4.7 Diskretisering af Regulator

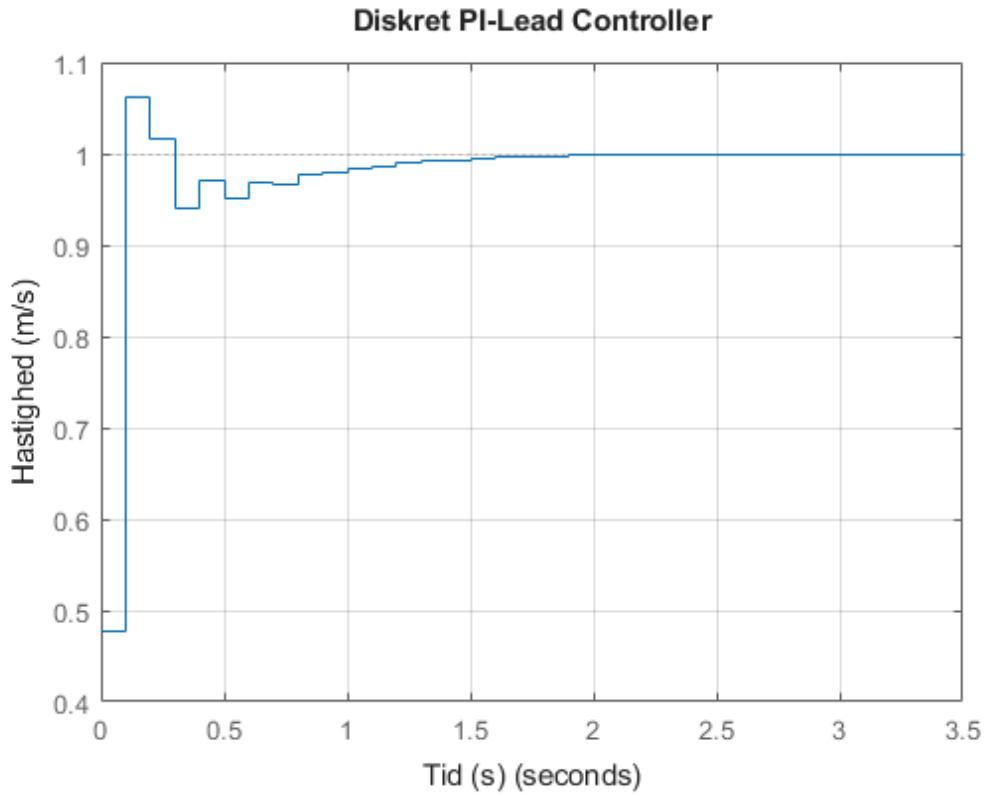
For at diskritisere regulatoren skal en sample-rate  $T_s$  vælges. For dette projekt blev  $T_s = 0.1$  valgt, svarende til 10 hastighedskommandoer pr. sekund. Derefter skal funktionen

flyttes fra det kontinuere  $s$ -domæne, til det diskrete  $z$ -domæne. Dette gøres via MATLABs `c2d` funktion med argumentet 'tustin', som er tilsvarende til at anvende den bilineære transformation

$$s \approx \frac{2}{T_s} \frac{1 - z^{-1}}{1 + z^{-1}}$$

Dette anvendes på den kontinuerlige regulator  $C_{ol}(s)$  for at udlede den diskrete regulator  $C_{ol}(z)$ :

$$C_{ol}(z) = \frac{u(z)}{e(z)} = \frac{23.05 z^2 - 32.01 z + 11.07}{z^2 - 0.3182 z - 0.6818} \quad (4.5)$$



Figur 9: Diskret PI-Lead af den optimale kontinuerlige PI-Lead.

Diskretiseringen danner et overshoot på 6.23%, hvilket skyldes bl.a. at den bilineære transformation kun er en approksimation. Før dette kan blive direkte implementeret i C-koden, skal den omskrives til en differensligning:

$$u(z)(z^2 - 0.3182z - 0.6818) = e(z)(23.05z^2 - 32.01z + 11.07)$$

hvilket medfører:

$$\begin{aligned} u[k] - 0.3182 \cdot u[k-1] - 0.6818 \cdot u[k-2] &= 23.05 \cdot e[k] - 32.01 \cdot e[k-1] + 11.07 \cdot e[k-2] \\ \Rightarrow u[k] &= 0.3182 \cdot u[k-1] + 0.6818 \cdot u[k-2] + 23.05 \cdot e[k] - 32.01 \cdot e[k-1] + 11.07 \cdot e[k-2] \end{aligned}$$

hvor:

$$\begin{aligned} u[0] &= u[k] \\ u[1] &= u[k-1] \\ u[2] &= u[k-2] \end{aligned}$$

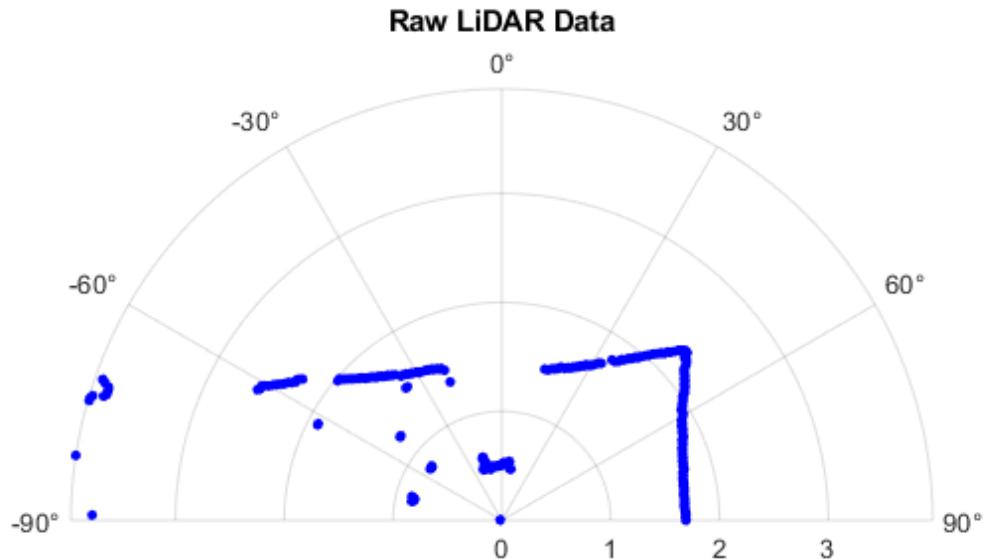
som resulterer i den endelige implanterbare differensligning

$$\underline{u[0] = 0.3182 \cdot u[1] + 0.6818 \cdot u[2] + 23.05 \cdot e[0] - 32.01 \cdot e[1] + 11.07 \cdot e[2]} \quad (4.6)$$

Det er  $u[0]$ , som er den hastighed  $v$ , der skal bruges til at udregne hjulenes individuelle hastigheder i ligning (3.5). Selvfølgelig skal idéen om en retning også indgå i hastighedsudregningen, som tager form af krumningen  $\kappa$ , og dette kræver at LiDAR dataen bliver tolket korrekt.

## 5 Centroid-Detektering via LiDAR data

Robotterne er udstyret med en LiDAR-sensor, der er monteret på forsiden af hver robot. Indtil den rå data produceret af sensoren er blevet fortolket, er den ikke særlig brugbar. Det er let for et menneske at tolke figur 10, og udledte at den blå linje til højre er en mur, og koncentrationen i midten er en robot placeret foran en døråbning. Programmet ved kun, at der er nogle datapunkter, men hvad de betyder, samt hvordan det skal tolkes, skal indføres i form af en algoritme, som tillader robotten at dekode punkternes betydning.



Figur 10: Rå LiDAR-sensor uden fortolkning.

Det er ret åbenlyst, at datapunkterne kan grupperes og tolkes som objekter. Eksempelvis så er datapunkterne, der ligger tæt på hinanden, og ligger i lige linje, nok højest sandsynligt et væglnærende objekt. Ligeledes kan man tolke en givet gruppe af datapunkter, som indeholder en specifik mængde af datapunkter, med en given densitet, som værende en robot. Dette kaldes clustering [Wik25]. Målet er at isolerer de datapunkter, som mest sandsynligt tilhører robotten.

### 5.1 Clustering

Når LiDAR-sensoren laver en skanning, sker det fra venstre mod højre. De 512 datapunkter gemmes i et array  $\mathbf{S}[i]$ ,  $i \in [0; 511]$ , hvor det første index  $\mathbf{S}[0]$  er ved  $-90^\circ$  og sidste  $\mathbf{S}[511]$  er  $90^\circ$ . Dermed er ligeud  $0^\circ$ . I virkeligheden er synsfeltet for LiDAR-sensoren  $184.32^\circ$ , hvilket medfører, at vinklen mellem hvert datapunkt er givet ved

$$\Delta\theta_i = \frac{184.32^\circ}{512} = 0.36^\circ$$

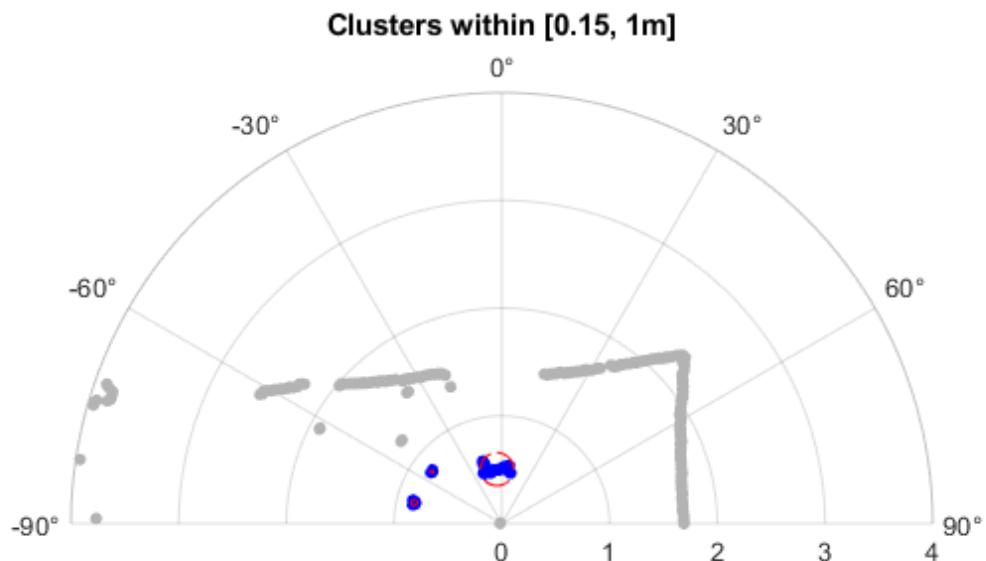
hvorfra det udledes, at for et givet index  $i$ , er vinklen

$$\theta_i = i \cdot 0.36 - 90^\circ \quad (5.1)$$

Det vil sige, at for et givet index  $i$  i arrayet  $\mathbf{S}$  gælder:

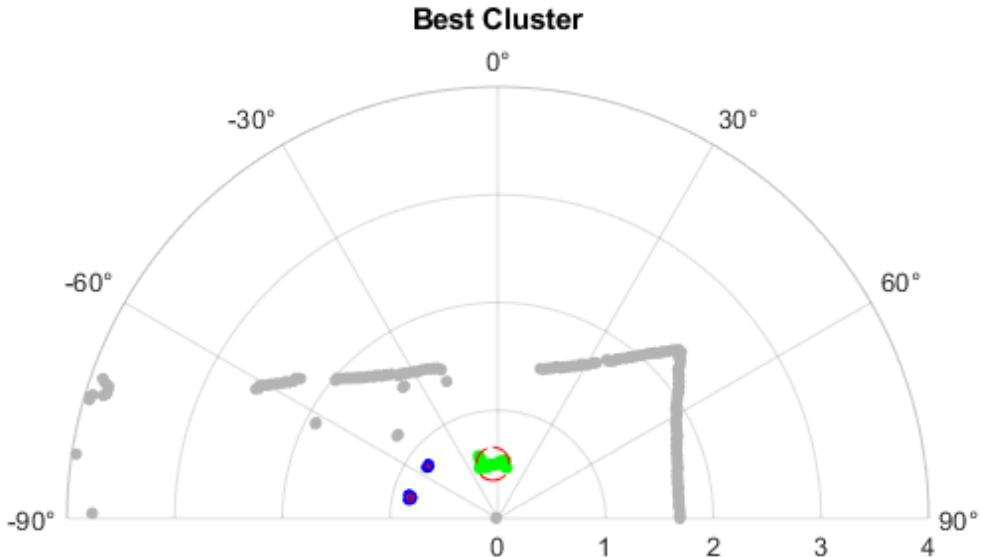
$$\begin{cases} \mathbf{S}[i] &= r_i \\ \theta_i &= i \cdot 0.36 - 90^\circ \end{cases} \quad i \in [0; 511]$$

En hurtigt måde at filtrere store mængder af overflødig data, sker ved at lave en kraftig antagelse baseret på distance. En nedre grænse på  $0.15m$ , og en øvre grænse på  $1m$  defineres som det tilladte interval.



Figur 11: Filtret data baseret på distance.

Med andre ord ignoreres datapunkter, hvis distance ikke er i intervallet  $(0.15m; 1m]$ . Denne antagelse er rimelig, da hvis et objekt befinder sig indenfor  $15cm$ , så er noget gået galt, og robotten burde i så fald udføre et nødstop. Ved at sætte en øvre grænse på én meter, er det med stor sandsynlighed den lige forankørende robot, som er opfanget, og ikke den foran igen. Dette er meget brugbart, da det i et sving er muligt, for f.eks. robot nummer 3 i konvojen, at fange robot nummer 1, når det skulle have været robot nummer 2, der skulle sigtes efter. Altså, at den opfanger en robot længere fremme end den skulle. Ved at indføre en øvregrænse formindskes sandsynligheden for denne situation.



Figur 12: Bedste Cluster (grøn) valgt baseret på algoritme.

Den næste filtering er baseret på antal indeks,  $\mathbf{S}[i], \mathbf{S}[i + 1] + \dots + \mathbf{S}[i + n]$ , da det er forventeligt, at en robot vil blive ramt af  $N$ -antal LiDAR-stråler, an på hvor tæt den er på sensoren. En større distance vil betyde færre punkter. Derfor defineres intervallet af datapunkter, som kan tilhører en robots cluster til  $N \in [20; 140]$ . Med denne implementation er hver følgerobot i stand til at identificerer en forankørende robot, men for at gøre implementningen lettere skal en såkaldt 'centroid' introduceres.

## 5.2 Centroid

En centroid er et enkelt punkt, som repræsenterer gennemsnitsdensitet af clusteret. For at udregne centroiden bliver hver punkt først transformeret til kartesiske koordinater:

$$\begin{aligned}\mathbf{x}_i &= r_i \cdot \cos(\theta_i) \\ \mathbf{y}_i &= r_i \cdot \sin(\theta_i)\end{aligned}$$

Herefter tages gennemsnittet af alle clusterets  $(x, y)$  koordinater for at få centroidens lokation,

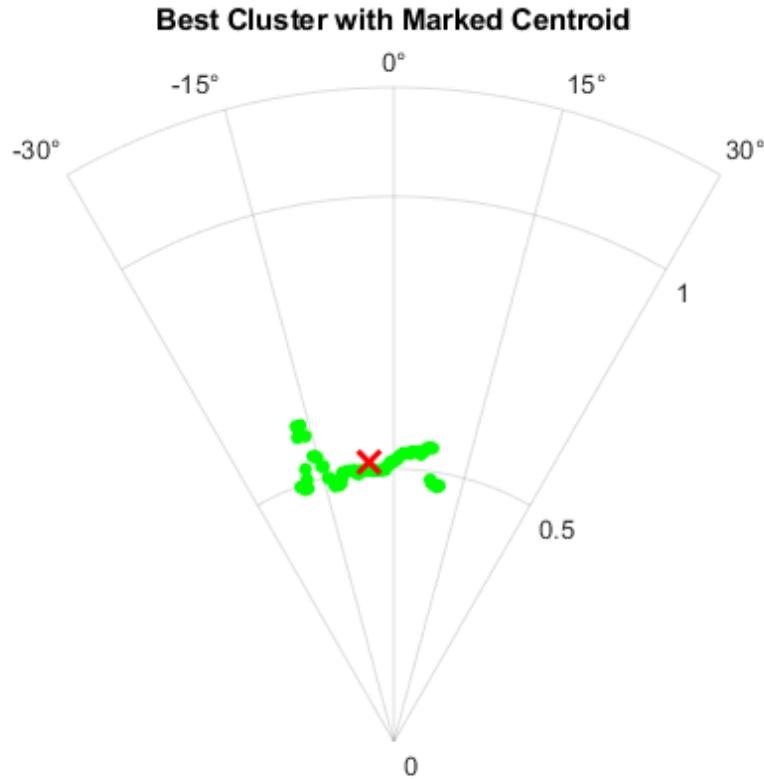
$$\begin{aligned}x_c &= \bar{\mathbf{x}} = \frac{1}{N} \sum_{i=0}^N x_i \\ y_c &= \bar{\mathbf{y}} = \frac{1}{N} \sum_{i=0}^N y_i \\ c_{x,y} &= (x_c, y_c)\end{aligned}$$

Centroidens lokation er defineret som gennemsnittet af  $\mathbf{x}$  og  $\mathbf{y}$ , og det er dette punkt, som følgerobotten skal køre efter.

I udregningen af krumningen indgår vinklen mellem retning af følgerobotten og centroiden er det værd at transformere til polære koordinater

$$\begin{aligned}c_{r,\theta} &= \left( \sqrt{x_c^2 + y_c^2}, \tan^{-1} \left( \frac{y_c}{x_c} \right) \right) \\ c_{r,\theta} &= (r, \theta)\end{aligned}$$

Den sidste filtrering forgår ved at logge alle fundne centroider i et array  $\mathcal{C}$ , og derfra vælge den centroid  $\mathcal{C}[n]$ , som minder mest om den tidligere udvalgte robot  $\mathcal{C}[n - 1]$ . I tilfældet af én valid centroid vælges denne altid.



Figur 13: Centroid udregnet baseret på samme data fra figur 12.

Hele algoritmen ses i algoritme 1

#### Algorithm 1 Simpel Centroid-Detektering

- 1: Filtre punkter baseret på distance og størrelse.
- 2: Gruppér valide punkter til et cluster.
- 3: Fjern clusters, der ikke opfylder  $N \in [20, 140]$ .
- 4: Udregn gennemsnitsvinkel og -distance.
- 5: Vælg de bedste clusters og udregn centroids.
- 6: Sammenlign centroids med forrige centroid, og vælg den, med mindst ændring.
- 7: **return** Centroid vinkel og distance  $c = (r_c, \theta_c)$ .

Derefter er det næste skridt at implementere de forskellige aspekter i form af kode, og bruge empiriske metoder til at udlede en endelig implementation.

---

## 6 IMPLEMENTERING AF HASTIGHEDSREGULATOR & CENTROID-DETEKTERING

---

## 6 Implementering af Hastighedsregulator & Centroid-dtektering

I teorien kan koden direkte implementeres med algoritmen 1 og ligning (3.5) med  $v = u[0]$  fra ligning (4.6),

$$v_R(t) = u[0] \cdot \left(1 + \frac{\kappa \cdot b}{2}\right)$$
$$v_L(t) = u[0] \cdot \left(1 - \frac{\kappa \cdot b}{2}\right)$$

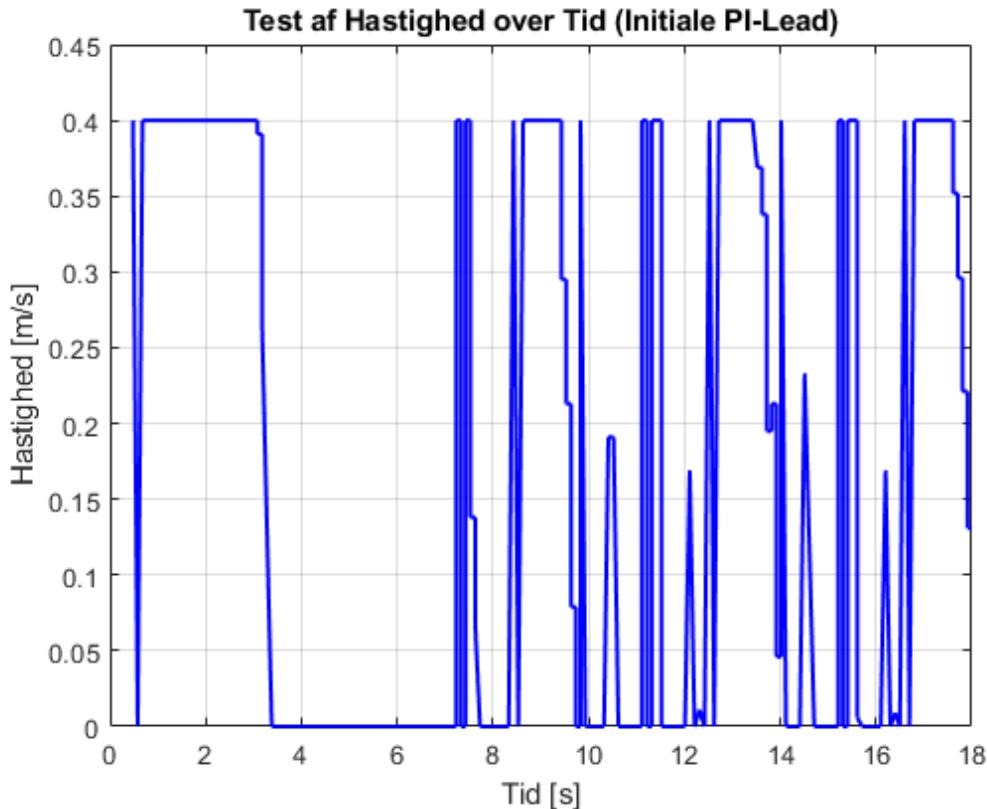
hvor

$$u[0] = 0.3182 \cdot u[1] + 0.6818 \cdot u[2] + 23.05 \cdot e[0] - 32.01 \cdot e[1] + 11.07 \cdot e[2]$$
$$\kappa = 2 \cdot \frac{\sin(\theta_{err})}{L}$$
$$\theta_{err} = \theta_c - \theta_h$$

I realiteten har dette dog ikke den ønskede opførelse, og kræver derfor nogle justeringer, som er fundet via empiriske forsøg.

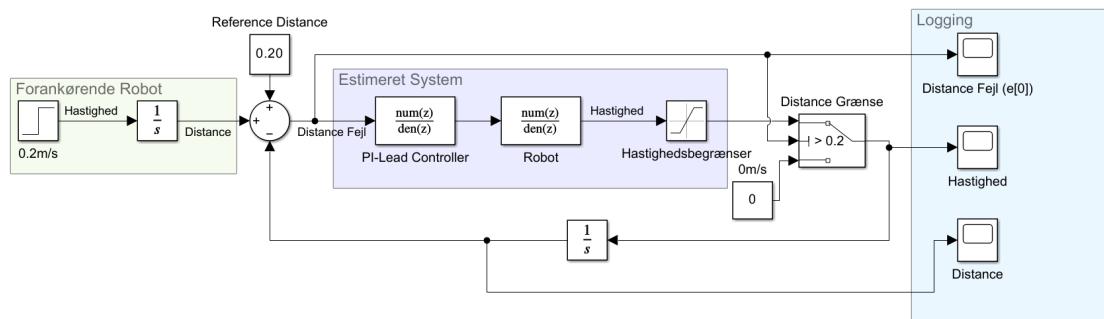
### 6.1 Justering af hastighedsregulator

I en af de første testimplementeringer af hastighedsregulatoren blev en følgerobot placeret bagved en førerrobot, som bevægede sig med  $0.2m/s$ . Følgerobotten havde en begrænset maksimalhastighed på  $0.4m/s$ . Det var forventet at følgerobotten skulle nå en topfart på de  $0.2m/s$ , og vedligeholde hastigheden mens den bevægede sig bag førerobotten, men som figur 14 tydeligt viser, var dette ikke tilfældet.



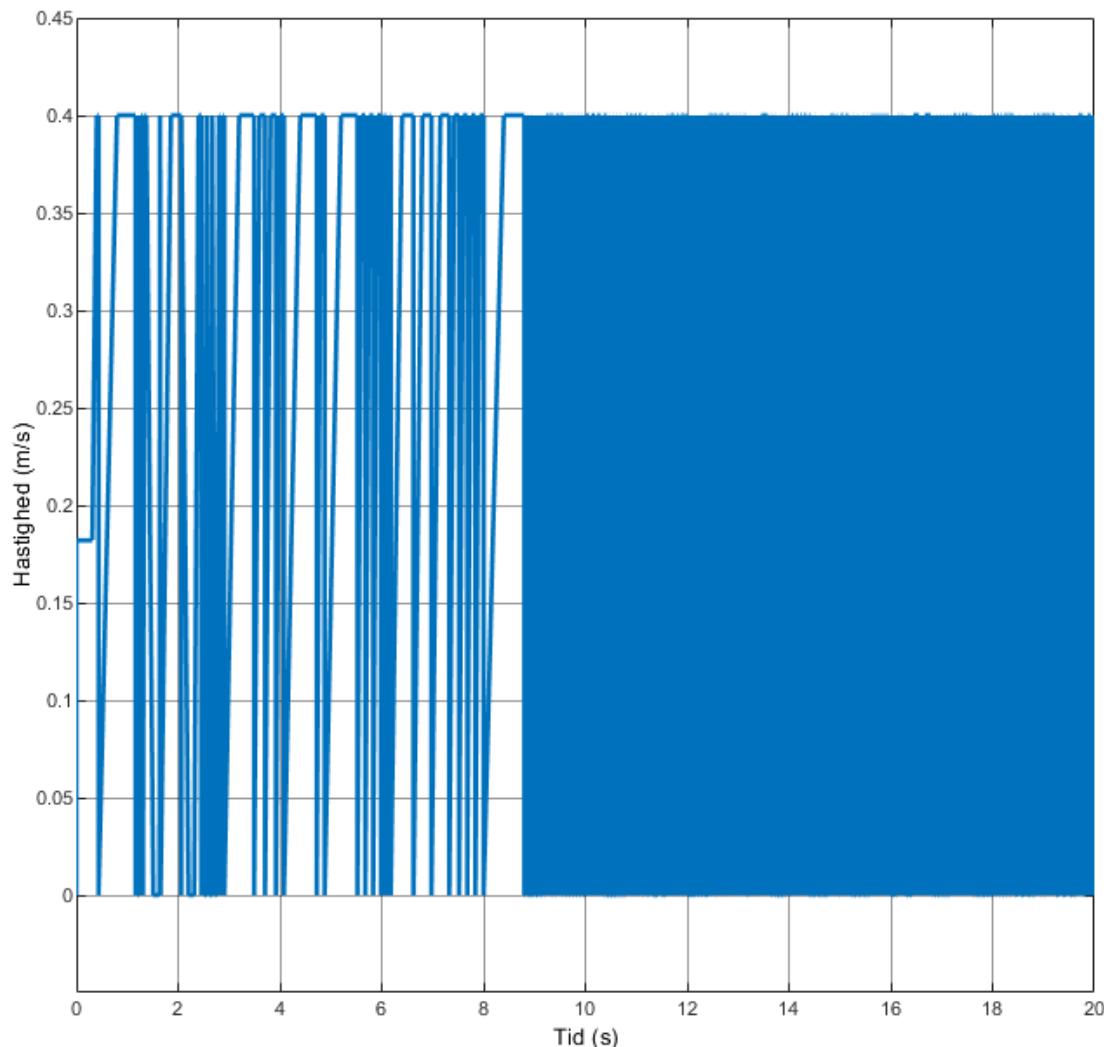
Figur 14: Oscillerende hjulhastigheder af følgerobot, der følger en robot, som kører  $0.2m/s$ .

De meget voldsomt oscillerende hjulhastigheder skyldes, at hastighedsregulatoren indtil nu kun er designet ud fra et hastighedsperspektiv. Hastighedsregulatoren blev designet uden at tage højde for sensoren . Modellen udvides ved at opstille en model, der også tager højde for hastighedsbegrænsningerne, samt distance kriteriet, hvilket gør det muligt at simulerer situationen.



Figur 15: Modellering af systemet som et blokdiagram i Simulink.

Figur 15 viser en model af forsøget, hvor den forankørende robot er modelleret som en stepfunktion integreret. Det estimeret system er fortsat uændret, men en 'switch' er implementeret på robottens output, da hvis afstanden mellem robotterne er mindre en  $20\text{cm}$  så sættes hastigheden til  $0\text{m/s}$ . Yderligere integreres hastigheden, da dette giver positionen. Det forstår således, at forskellen i tilbagelagt strækning er proportional med afstanden mellem de to robotter. Modellen påtrykkes med stepfunktion til  $0.2\text{m/s}|_{s=2}$ .



Figur 16: Modellering af oscillerende hjulhastigheder.

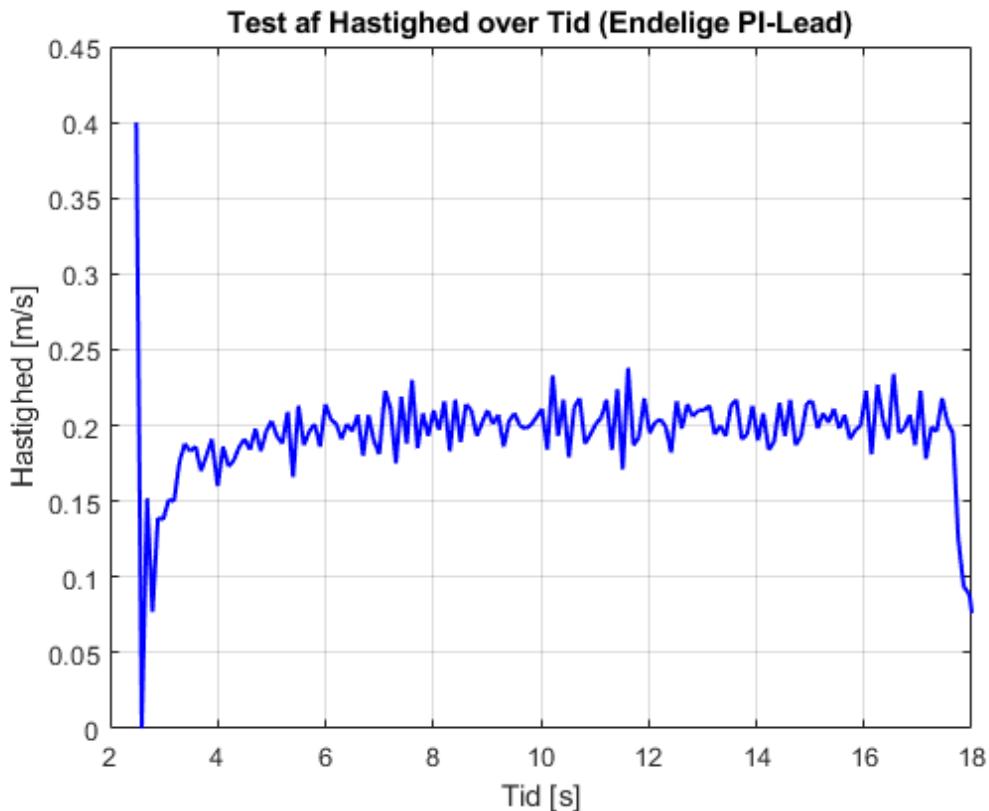
Det er en tilnærmelsesvis god modellering af systemet, men afviger på nogle punkter. Såsom at den over oscillerer, men dette skyldes, tildels at robotten i virkeligheden har inertie og kan lave hjulskred, som gør at hastigheden ikke går til  $0\text{m/s}$  på et øjeblik, men den aftager over en kort periode. Problemets er, at regulatoren forsøger på at justere hastighed til  $0.4\text{m/s}$ , uden at tage højde for distancen til den forankørende.

Hvilket resulterer i den kommer for tæt på den forankørende robot, og dermed aktiverer nødstopsystemet, indtil afstanden bliver større end 20cm.

I den endelig implementering er regulatoren implementeret som differensligningen (4.6), og dermed giver det mening at ændre på denne. Det er alene ikke nok at ændre gainen, da det er tydeligt, at der er integration-windup. Heldigvis kan både gainen, og selve regulatoren ændres ved at gange en konstant på differensligningen. Ved empirisk at køre forsøget et par gange med forskellige faktorer, blev 0.3 den endelig værdi:

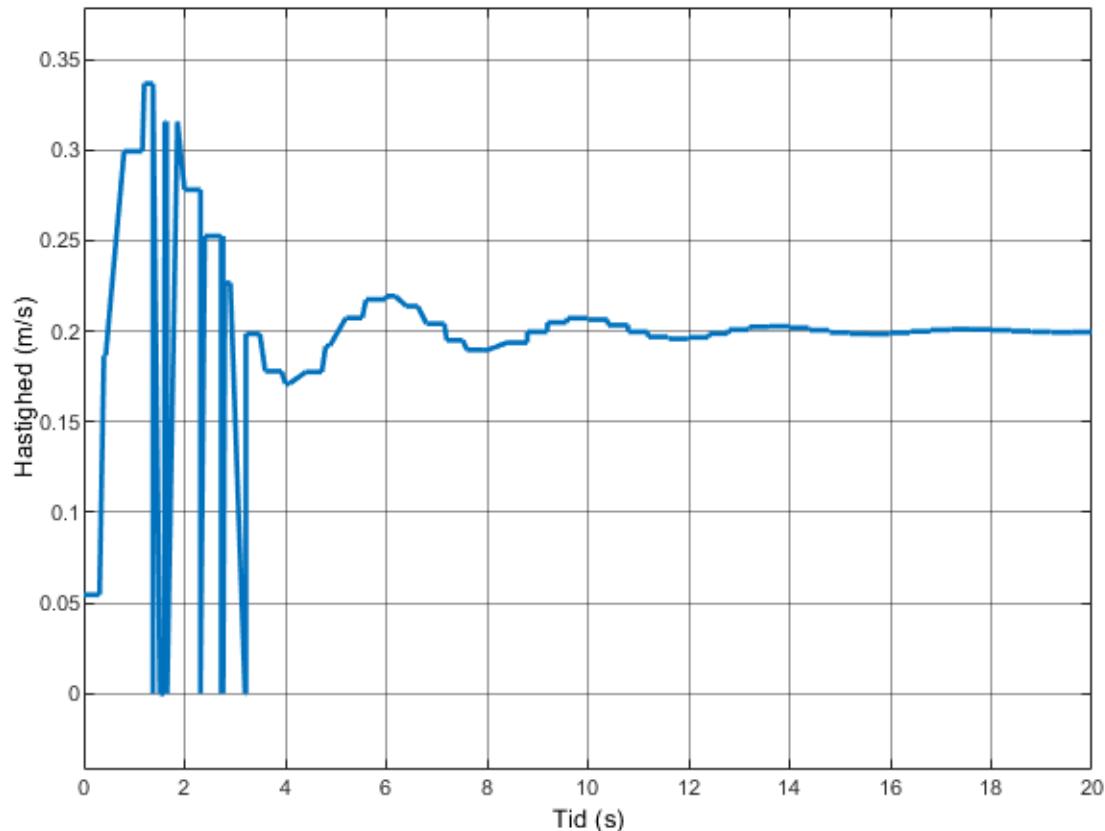
$$\begin{aligned}
 u[0] &= 0.3 \cdot (0.3182 \cdot u[1] + 0.6818 \cdot u[2] + 23.05 \cdot e[0] - 32.01 \cdot e[1] + 11.07 \cdot e[2]) \\
 \Rightarrow u[0] - 0.3 \cdot (0.3182 \cdot u[1] + 0.6818 \cdot u[2]) &= 0.3 \cdot (23.05 \cdot e[0] - 32.01 \cdot e[1] + 11.07 \cdot e[2]) \\
 \Rightarrow 1 - 0.3 \cdot (0.3182 \cdot z^{-1} + 0.6818 \cdot z^{-2})u(z) &= 0.3 \cdot (23.05 - 32.01 \cdot z^{-1} + 11.07 \cdot z^{-2})e(z) \\
 \Rightarrow \frac{u(z)}{e(z)} &= \frac{0.3 \cdot (23.05 - 32.01 \cdot z^{-1} + 11.07 \cdot z^{-2})}{1 - 0.3 \cdot (0.3182 \cdot z^{-1} + 0.6818 \cdot z^{-2})} \\
 \Rightarrow \frac{u(z)}{e(z)} &= \frac{0.3 \cdot (23.05z^2 - 32.01 \cdot z + 11.07)}{z^2 + 0.3 \cdot (-0.3182 \cdot z - 0.6818)} \\
 \Rightarrow \frac{u(z)}{e(z)} &= \underline{\underline{\frac{6.915z^2 - 9.603z + 3.321}{z^2 - 0.09546z - 0.20454}}}
 \end{aligned}$$

For at validere den nye regulator blev samme forsøg blev gentaget som for den gamle. Resultatet ses på figur 17.



Figur 17: Forbedret hjulhastigheder af følgerobot, der følger en robot, som kører  $0.2\text{m/s}$ .

Det ses at regulatoren først overreagerer med en skarp stigning. Dette skyldes at fejlen  $e[0]$  (afstanden til forankørende) går fra at være større end  $20\text{cm}$ , samt at den har en hastighed på  $0\text{m/s}$ . Dermed reagerer regulatoren voldsomt. Dette stemmer overens med hvad dens tilsvarende simulering visser på figur 18.

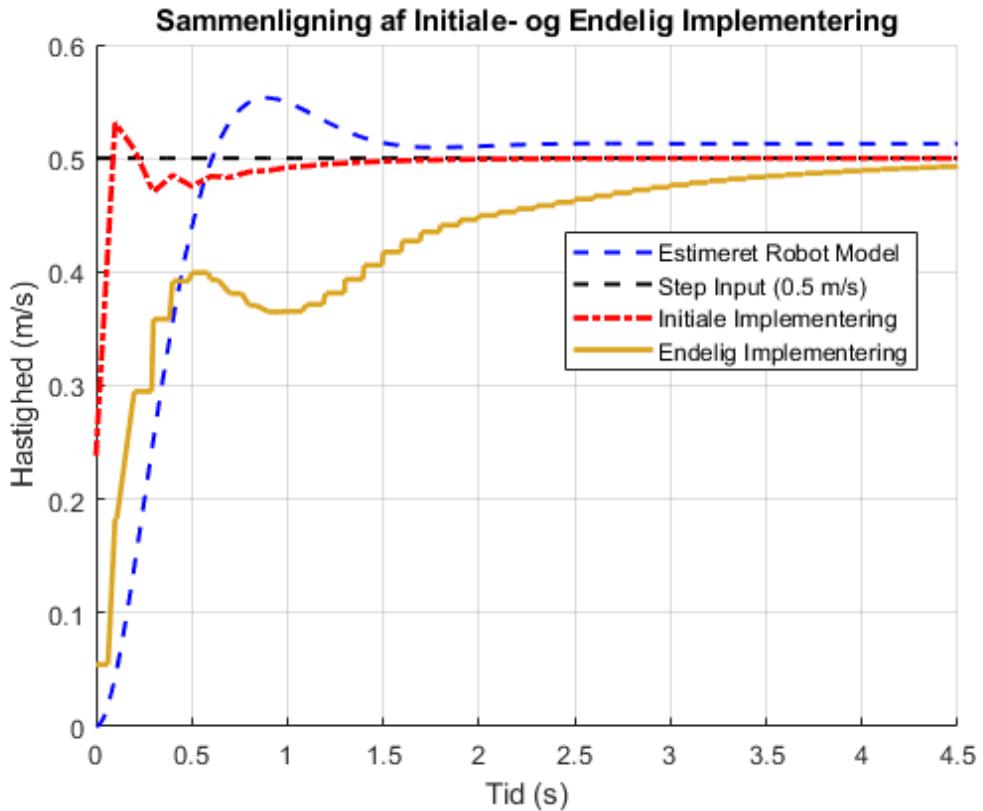


Figur 18: Modellering af den endelige implementering i samme scenarie.

Igen skyldes afvigelserne fysiske parametre som modellen ikke tager højde for. Det ses dog, at modellen og forsøget stemmer overens, og det eneder med, at robotten følger hastigheden på den forankørende, hvilket er det ønskede resultat.

#### 6.1.1 Steprespons af hastighedsregulator

Det er værd at analysere hvordan den nye implementering burde opføre sig i teorien, og sammenligne den med systemets originale respons, samt den initiale implementation udledt i afsnit 4.7.



Figur 19: Sammenligning af den estimeret model af robotten, den første implementering, og den endelige implementering

På figur 19 ses det tydeligt, at den endelige implementation har en markant længere rise-time sammenlignet med den initiale implementation og den estimeret model for systemet. Der bliver offeret en hurtigt settling-time i bytte for en mere glat aftagende respons. Der er i teorien plads til yderligere optimering, men da regulatoren opfylder kravene implementeres den.

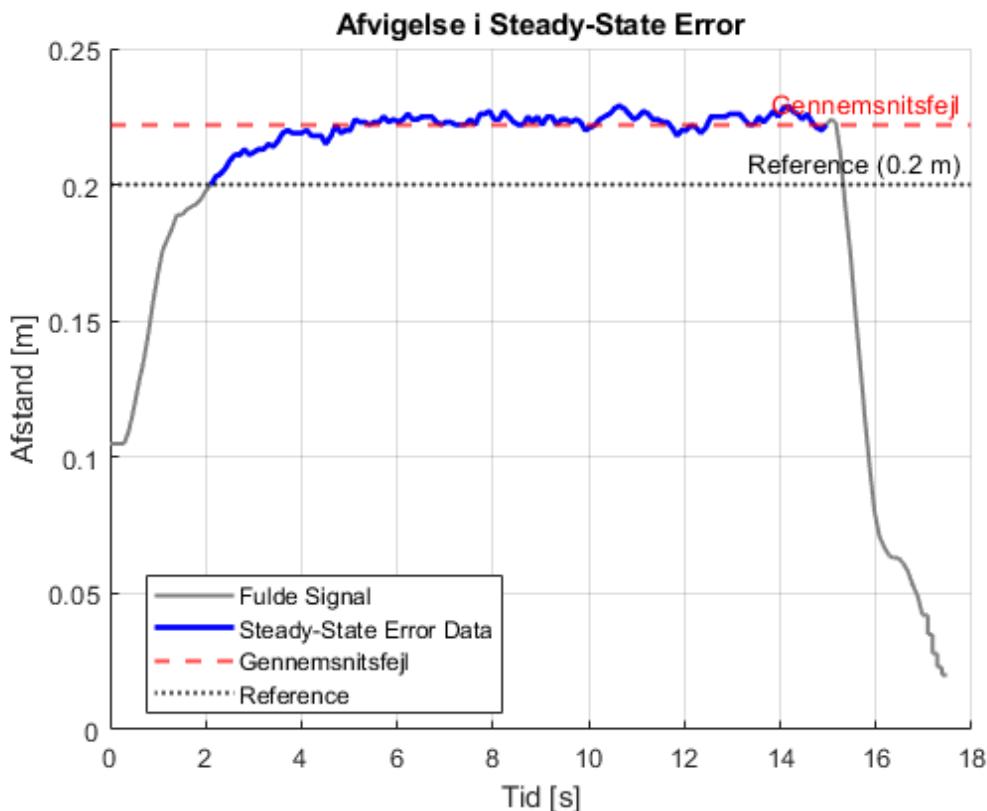
### 6.1.2 Steady-State Error

Det er af interessa at sammenligne den teoretiske models Steady-State Error, med robottens Steady-State Error. Den teoretiske Steady-State Error er på omkring  $e_{ss} \approx 0.05 = 5\%$ , hvilket er inden for rimelighedens grænser. I den reelle implementation er det LiDAR-sensoren i feedback-loopet, som vurderer afstanden til den forankørende. Fra samme forsøg som producerede data til figur 17, blev afstandsfejlen  $e[0]$  desuden målt. Der blev kun kigget på det tidsrum hvor forsøget nåede en endelig værdi, hvorfra

gennemsnitsfejlen for Steady-State erroren udregnes

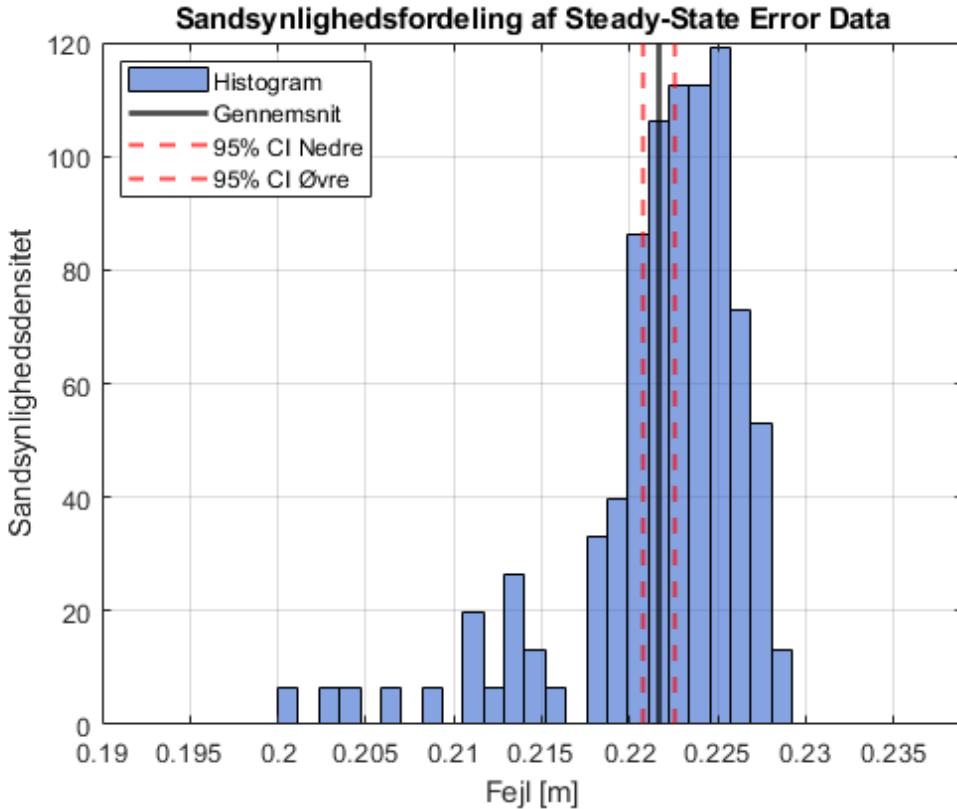
$$\bar{e} = \frac{1}{N} \sum_{i=0}^N e_0[i] \approx 0.2217m$$

Hvilket giver anledning til resultatet i figur 20.



Figur 20: Plot af den gennemsnitlige fejl, det udvalgte data, og reference distancen.

Der er en fejl på ca. 10.85% i forhold til den ønskede afstand til den forankørende. Figur 21 viser 95%-konfidensintervallet for gennemsnitsfejlen  $\bar{e} = 0.2217m$ . Det ses desuden, at der er en bias på  $0.0217m$ , hvilket er en acceptable bias, da det er tilsvarende til ca. 2% hvilket er inden for de 3% [Hok25] præcision som LiDAR-sensoren har. Hvilket betyder at fejlen er sandsynligvis konsistent, da histogrammet viser en tydelig koncentration omkring biasen.



Figur 21: Histogram af  $e[0]$ .

Udfra figur 17 og figur 20 konkluderes det, at den reelle afvigelse fra teorien skyldes fejlkilder, såsom hjulslip, da dækkenne på SMR robotterne er af stof, og dermed har en laver friktionskoefficient, end hvis de var af gummi, osv. For at opsummere har den reelle implementation en tilfredsstillende respons på det givne input, og en acceptabel afvigelse fra den ønskede afstand.

## 6.2 Justering af distance regulering

Den nuværende implementering resulterer i at systemet reagere med det samme, når førerrobotten begynder at dreje, samt at den drejer for skarpt. Dette er to forskellige problemer, men de er sammenkoblet. I udregningen for krumningen  $\kappa$  introduceres der nu 'lookahead'-distancen  $L_d = 0.8m$  i stedet for kordens længde  $L$ . Idéen er at antage, at korden er konstant. Dette medfører implicit, at regulering af distancen via hastigheds- og retningsregulatoren bliver separeret. Empiriske forsøg viser at  $0.8m/s$  er en god værdi. Derudover så introduceres  $\kappa_{raw}$ , der er defineret som

$$\kappa_{raw} = 2 \cdot \frac{\sin(\theta_{err})}{L_d}$$

samt  $\tilde{\kappa}$ , som er den filtrerede krumning. Idéen er at indfører logik som tillader den rå  $\kappa$  at blive udtrykt for mere aggressive sving, og  $\tilde{\kappa}$  for glatte sving.

$$\tilde{\kappa}_n = \begin{cases} \kappa_{raw}, & |\theta_{err}| > 15^\circ \\ 0.7 \cdot \tilde{\kappa}_{n-1} + 0.3 \cdot \kappa_{raw}, & otherwise \end{cases} \quad (6.1)$$

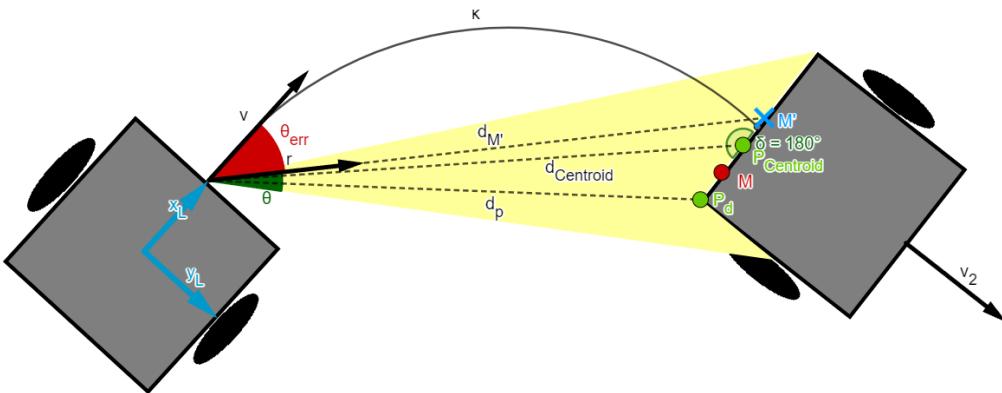
hvor  $\tilde{\kappa}_n$  er den nuværende krumning, og  $\tilde{\kappa}_{n-1}$  er den tidligere udregnet krumning. Dette er et simpel lowpass-filter som gør svingene mere glatte.

### 6.2.1 Hjørneskæring

Selv med disse implementeringer er der stadig et problem med hjørneskæringer. Robotten reagere stadig for tidligt, idet den drejer før end den burde. For at undgå dette, implementeres en buffer, som har til formål at forsinke hvilken centroid robotten reagere på. I stedet for at reagere på den nuværende centroid  $C[n]$ , så reagere robotten på  $C[n - 5]$ . Dette tillader følgerobotten at køre frem til den position, hvor førerrobotten befandt sig, og derefter dreje, hvilket medfører, at den undgår hjørneskæring.

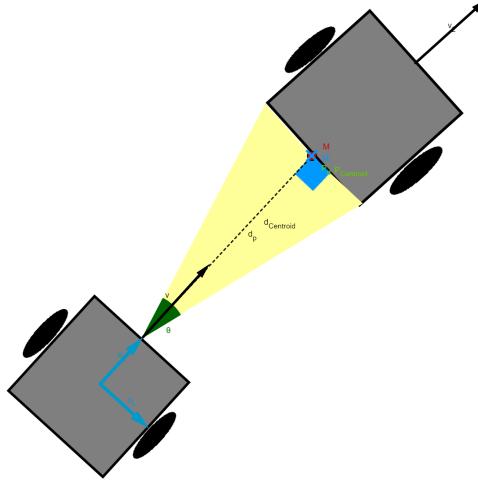
### 6.2.2 Centroid projektion

Antallet af punkter i et cluster samt clusterets position er proportional med, hvor stor en del af førerrobotten der kan ses, hvilket igen afhænger af, hvor meget de to robotters retningsvektorer er roteret i forhold til hinanden. Dette medfører, at det er muligt for centroiden, som ideeltset burde være centreret på bagsiden af robot og være den korteste distance mellem de to robotter, at være forskudt fra midtpunktet (se figur 22).



Figur 22: Illustration af centroid forskydning.

Med andre ord, er centroiden forskudt fra midtpunktet af robotten, i retningen mod punktet  $P_d$ , som er den korteste afstand mellem de to robotter. For at rette denne fejl findes midtpunktet  $M$  mellem centroiden  $P_{centroid}$  og punktet  $P_d$ , hvor punktet  $M$  roteres  $180^\circ$  omkring  $P_{centroid}$ , hvilket giver punktet  $M'$ . Det betyder, at det er punktet  $M'$  som robotten skal køre efter i stedet for.



Figur 23: Situationen hvor  $M' = P_d = P_{Centroid}$ , robotterne kører på samme linje.

I tilfældet hvor  $\|P_{Centroid} - P_d\| \rightarrow 0$ , altså at centroiden er tæt på være det nærmeste punkt i clusteret, da gælder  $M' = P_d = P_{Centroid}$ . Dette er tilfældet hvor deres hastighedsvektorer er parallelle,  $\mathbf{v}_1 \parallel \mathbf{v}_2$ , og ligger på samme linje  $\mathbf{v}_1 = k \cdot \mathbf{v}_2$  (se figur 23). Det er dog tydeligt, at der stadig er en fejlmargin på det reelle centrum af den forankørende robot (som er lokeret på linjen udspændt af dens tilhørende hastighedsvektor  $v$ ), og den projekteret centroid  $M'$ . Herved forefindes hjørneskæring stadig, men markant mindre.

### 6.3 Søgeområde

Idet robotterne ikke er i stand til hoppe fra et punkt til et andet, giver det mening at vælge det cluster, der minder mest om det forrige. Det er forventeligt, at afstanden og vinklen fra en centroid til den næste centroid for samme robot, ikke burde ændre sig markant. På samme vis som med krumningen  $\kappa$  i afsnit 6.2 introduceres en bias hvis formål er at øge chancen for, at en robot vælger den rigtige centroid. Set fra robottens eget koordinatsystem er ligeud  $\theta = 0^\circ$ , derfor er vinklen til den forankørende robot  $\theta_{err} \in [-90^\circ; 90^\circ]$ . Der defineres nu vinklen  $\phi$ , som er den vinkel, der søges med

$$\phi_n = \begin{cases} 0.7 \cdot \phi_{n-1} + 0.3 \cdot \theta_{err}, & |\theta_{err}| > 10^\circ \\ 0.9 \cdot \phi_{n-1} + 0.1 \cdot \theta_{err}, & otherwise \end{cases}$$

hvor  $n$  angiver den nuværende værdi. Der implementeres to forskellige lowpass filtre, hvis formål er at justere og begrænse, hvor det er forventet at finde den forankørende

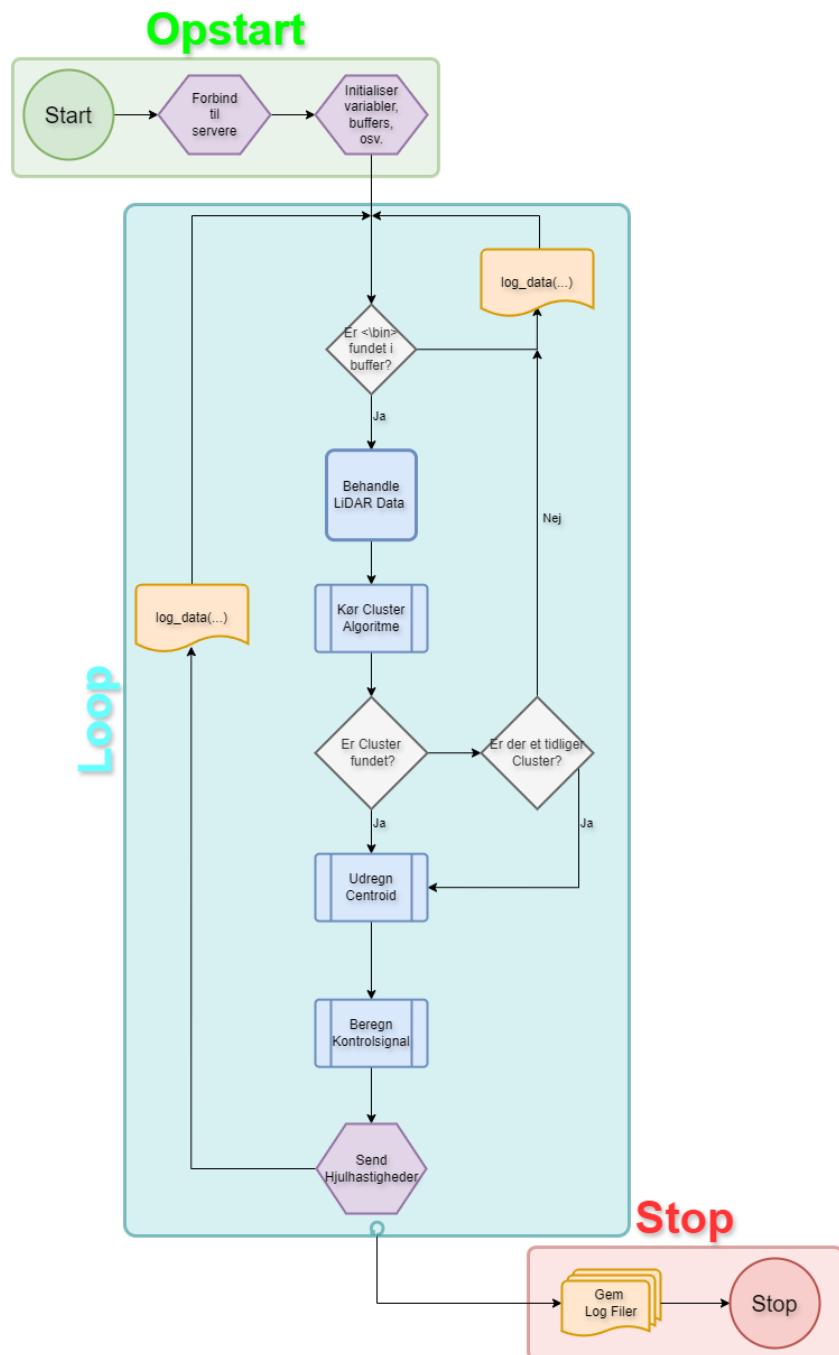
robot. Det er forventeligt, at  $\theta_{err}$  ikke har store svingninger eller markante ændringer. Det er dog en realitet, at centroidsene har usikkerheder, der kan opfattes som støj. Derfor filtres dette inden den bliver ført videre.  $\phi$  bliver indført som et ekstra trin i algoritme 1, således at  $|\theta_{cluster} - \phi| \approx 0$  er fortrukne.

#### 6.4 Fallback

Trods alle disse justeringer, er det muligt for en følgerobot at miste den forankørende robot. Dette kan skyldes skarpe sving, der får cluster-mængden til at falde under 20 datapunkter, at der er objekter tæt på den forankørende så cluster-logikken inkluderer dette objekts punkter som en del af robottens osv. I de tilfælde bliver **fallback** logikken anvendt, hvilket siger, 'anvend den sidst kendte retning og vinkel af den forankørende robot, hvis der er i 5 frames ikke bliver fundet nogle valide centroids'. Denne data bliver hentet fra samme buffer som i afsnit 6.2.1. Hvis en ny valid centroid bliver fundet, så kører robotten efter denne.

#### 6.5 Endelig implementering

For at opsummere den endelige implementering af følgerobotternes logik ses der på figur 24.



Figur 24: Flowchart af C-programmet, der styrer følgeroboterne.

For at udregne hjulhastigheder anvendes følgende formler, som også tager højde for krumningen af banekurven:

$$v_R(t) = u[0] \cdot \left(1 + \frac{\tilde{\kappa}_n \cdot b}{2}\right)$$

$$v_L(t) = u[0] \cdot \left(1 - \frac{\tilde{\kappa}_n \cdot b}{2}\right)$$

hvor

$$u[0] = 0.09546 \cdot u[1] + 0.20454 \cdot u[2] + 6.915 \cdot e[0] - 9.603 \cdot e[1] + 3.321 \cdot e[2]$$

$$\tilde{\kappa}_n = \begin{cases} \kappa_{\text{raw}}, & |\theta_{\text{err}}| > 15^\circ \\ 0.7 \cdot \tilde{\kappa}_{n-1} + 0.3 \cdot \kappa_{\text{raw}}, & \text{otherwise} \end{cases}$$

$$\kappa_{\text{raw}} = 2 \cdot \frac{\sin(\theta_{\text{err}})}{L_d}$$

$$\theta_{\text{err}} = \theta_c - \theta_h$$

og for at finde en centroid ud fra LiDAR-dataen, bruges følgende algoritme

---

**Algorithm 2** Implementeret Centroid-Detektering
 

---

- Filtre LiDAR-punkter baseret på afstand  $r \in [0.15, 1.0]$ .
- 2: Gruppér sammenhængende valide punkter til clusters.
  - Fjern clusters hvor antallet af punkter  $N \notin [20, 140]$ .
  - 4: **for** hvert gyldigt cluster **do**
    - Udregn centroidens vinkel  $\theta_c$  og afstand  $r_c$  som gennemsnit af punkterne.
    - 6: Find det punkt  $P_d$  med korteste afstand i clusteret.
    - Beregn midtpunkt  $M$  mellem  $P_d$  og centroiden  $P_{\text{centroid}}$ .
    - 8: Rotér  $M$   $180^\circ$  omkring  $P_{\text{centroid}}$  for at få projektionen  $M'$ .  
Anvend  $M'$  som ny centroid.
  - 10: **end for**
  - Evaluér hver centroid i forhold til tidligere  $\phi_{n-1}$  og vælg den nærmeste i vinkel:  
 $|\theta_c - \phi_n| \approx 0$ .
  - 12: Hvis ingen centroid fundet, og færre end 5 frames siden sidst: anvend sidst kendte  $c = (r_c, \theta_c)$ .
  - Hvis mindre end 5 frames uden valid centroid: aktiver fallback med fast  $c_{\text{last}}$ .
  - 14: **return** Centroid  $c = (r_c, \theta_c)$  for styring.
- 

hvor

$$\phi_n = \begin{cases} 0.7 \cdot \phi_{n-1} + 0.3 \cdot \theta_{\text{err}}, & |\theta_{\text{err}}| > 10^\circ \\ 0.9 \cdot \phi_{n-1} + 0.1 \cdot \theta_{\text{err}}, & \text{otherwise} \end{cases}$$

Med det implementeret i C-kode, skal programmet nu afprøves og vurderes.

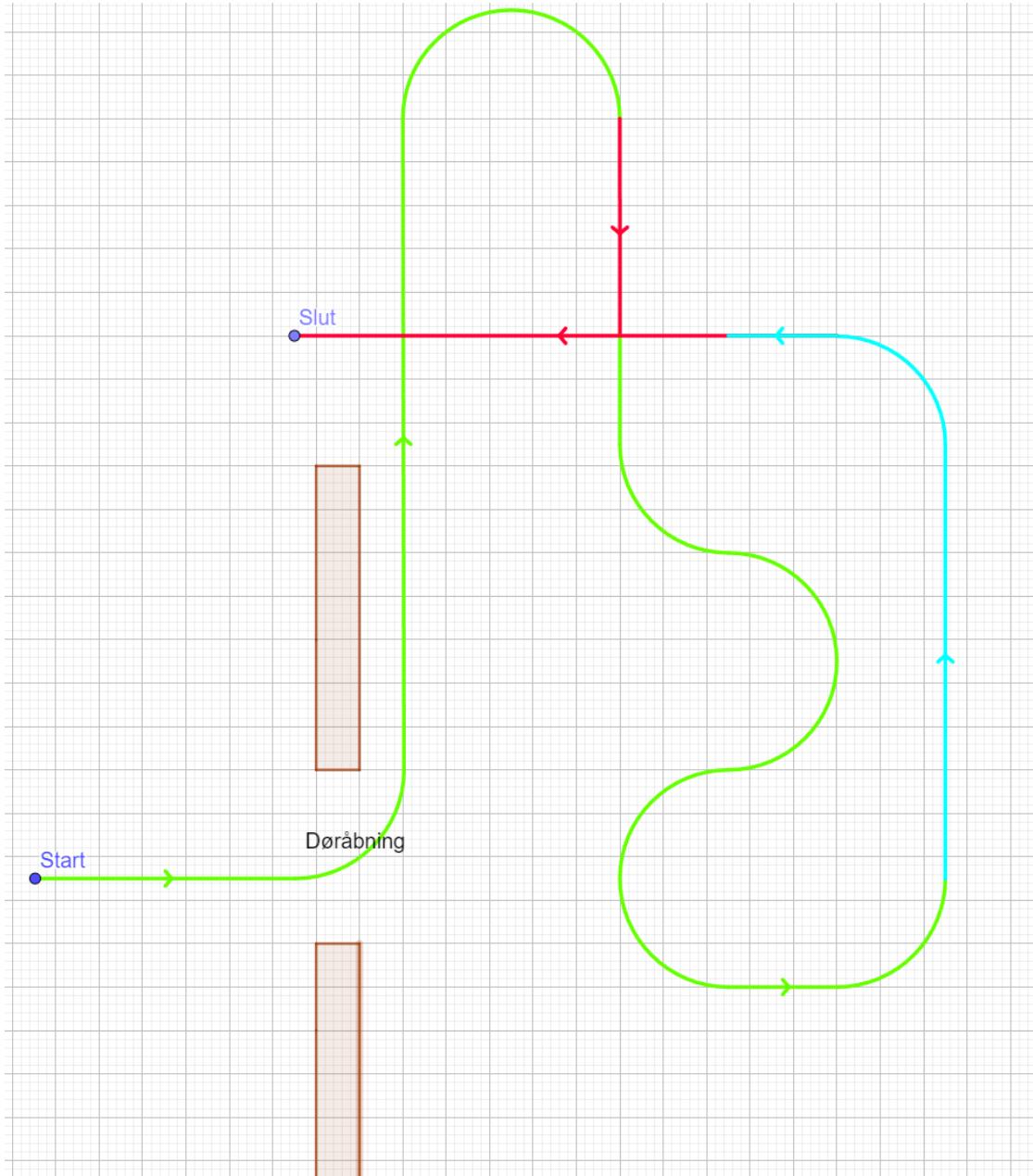
## **7 TEST AF IMPLEMENTERING**

---

### **7 Test af Implementering**

For at vurdere hvor godt programmet fungerer, er det essentielt at beskrive forskellige scenarier. Til testformål blev en forudprogrammeret rute bestemt til førerrobotten. Programmet er deterministisk, hvilket betyder, at det kan genskabes og forudsiges på forhånd. Figur 25 illustrerer den bane som førerrobotten kørte.

## 7 TEST AF IMPLEMENTERING



Figur 25: Ruten som førerrobotten kører, hvor farverne indikerer hastighederne (rød = $0.4m/s$ , grøn = $0.3m/s$ , blå = $0.2m/s$ ), hvor pilene indikerer kørselsretningen.

Idéen er at teste følgeprogrammet i forskellige scenerier, såsom

- Dreje om hjørner, hvor det er muligt for en centroid at blive tabt.
- Accelerer på en kort afstand, som tester følgeroboternes evne til at vedligeholde afstanden til forankørende, og så pludseligt skulle bremse.

- Slangekørsel, som tester programmets evne til ikke at skære hjørner, samt centroid vedligeholdelse.
- Tilsidst en slutspurt, som afprøver følgeroboternes evne til at dreje, og så skulle accelererer for at holde afstanden.

Forsøget vil blive gentaget tre gange, med formålet at teste robusthed. I øvrigt filmes slangekørslen ovenfra for at sammenligne den indbyggede odometri med en uafhængig måling. Forsøgsopstillingen var således at SMR11 var førerrobotten, og derfra gik rækken som SMR8, SMR15, og bagerst SMR6.



Figur 26: Start position af konvojen.

## 7.1 Resultater

Forsøget blev udført i alt tre gange, med robotterne i samme position i konvojen, SMR11 som førerrobot, efterfulgt af SMR8, SMR15, og tilsidst SMR6. I dette afsnit bliver resultaterne af forsøgene fremført.

### 7.1.1 Odometri

Da datasættene er store, gives et kort resumé af strukturen af dataen. Tabel 1 og 2 er dataen for følge-robotterne, og tabel 3 er den log-fil som førerrobotten laver når en SMR-CL kommando log "\$...\$" [AR04] er givet.

Tabel 1: Indhold i `control_logX.csv` ( $X = 6, 8, 15$ )

Kolonnenavn	Beskrivelse
Time	Unix-tid i sekunder
v_L	Målt venstrehjulshastighed [m/s]
v_r	Målt højrehjulshastighed [m/s]
u0	Controllerens nuværende outputsignal
u1	Controllerens outputsignal fra forrige tidssteg
u2	Controllerens outputsignal fra to tidssteg før
e0	Fejl (reference - måling) ved nuværende tidssteg
e1	Fejl ved forrige tidssteg
e2	Fejl ved to tidssteg før

Tabel 2: Indhold i `full_logX.csv` ( $X = 6, 8, 15$ )

Kolonnenavn	Beskrivelse
Time	Unix-tid i sekunder
Angle	Vinkel til centroid (fx lederrobotten) i grader
Distance	Afstand til centroid [m]
Robot_X	Robottens lokale X-position [m]
Robot_Y	Robottens lokale Y-position [m]
Robot_Heading_deg	Robottens heading/vinkel i grader
Centroid_X	Centroidens lokale X-position [m]
Centroid_Y	Centroidens lokale Y-position [m]
ClusterSize	Antal punkter i detekteret centroid (LiDAR-Cluster)
Velocity_cmd	Kommando til robotens kørehastighed [m/s]

Tabel 3: Indhold i `log` (førerrobot)

Kolonne	Beskrivelse
x	Robottens x-position (globalt)
y	Robottens y-position (globalt)
heading	Robottens orientering i radianer
velocity	Robottens hastighed
time	Unix-tid

Et eksempel på et data fra `full_log` ses i tabel 4, der viser at centroiden bliver mistet og fundet igen, fordi den bevæger sig ind og ud af dens tilladte størrelse.

## 7.1 Resultater

## 7 TEST AF IMPLEMENTERING

Tabel 4: Data uddrag fra `full_log15.csv` (Test A)

Index		Time	Angle	Dist	X	Y	Heading	Centroid_X	Centroid_Y	ClusterSize	u[0]
...	...	...	...	...	...	...	...	...	...	...	...
33	1749039388.5461449623	-1.00	-1.000	0.000	0.000	0.000	0.000	-1.000	0.000	-1.000	-1.000
34	1749039388.6491029263	-1.00	-1.000	0.000	0.000	0.000	0.000	-1.000	0.000	-1.000	-1.000
35	1749039388.7419110431	-1.00	-1.000	0.000	0.000	0.000	0.000	-1.000	0.000	-1.000	-1.000
36	1749039388.8448870182	2.04	0.272	0.129	0.044	1.20	0.400	0.019	137	0.018	
37	1749039388.9477755941	1.98	0.272	0.133	0.044	1.50	0.407	0.019	136	0.023	
38	1749039389.0508489609	1.98	0.272	0.136	0.044	1.09	0.407	0.019	132	0.019	
39	1749039389.1521191571	1.98	0.272	0.136	0.044	1.09	0.407	0.019	132	0.019	
40	1749039389.2515385925	2.16	0.273	0.138	0.044	1.32	0.410	0.021	129	0.023	
41	1749039389.3520619869	1.98	0.280	0.138	0.044	1.32	0.418	0.020	127	0.070	
42	1749039389.4529974461	1.62	0.294	0.140	0.044	1.72	0.433	0.021	127	0.034	
43	1749039389.5526548916	1.62	0.294	0.140	0.044	1.72	0.433	0.021	127	0.034	
44	1749039389.6531980032	1.08	0.299	0.146	0.044	1.89	0.445	0.020	130	0.007	
45	1749039389.7533557415	1.08	0.305	0.155	0.045	1.89	0.457	0.021	132	0.054	
46	1749039389.8535940647	1.08	0.305	0.165	0.045	1.78	0.470	0.020	135	0.057	
47	1749039389.9536939500	0.56	0.305	0.173	0.045	1.78	0.470	0.014	139	0.005	
48	1749039390.0542719364	-1.08	0.298	0.180	0.006	1.78	0.478	0.014	139	0.005	
49	1749039390.1545969003	-0.07	0.275	0.187	0.006	1.78	0.462	0.014	-1	-1.000	
50	1749039390.2543632019	-0.56	0.275	0.191	0.006	1.78	0.466	0.013	-1	-1.000	
51	1749039390.2549479008	-0.42	0.275	0.191	0.006	1.83	0.466	0.013	-1	-1.000	
52	1749039390.3552249676	-0.30	0.275	0.193	0.006	1.78	0.468	0.013	-1	-1.000	
...	...	...	...	...	...	...	...	...	...	...	...

For alle forsøgene sluttede robotterne tilnærmelsesvist i samme position, som vist på figur 27, og for alle tre forsøg kørte robotterne i en pæn konvoj.



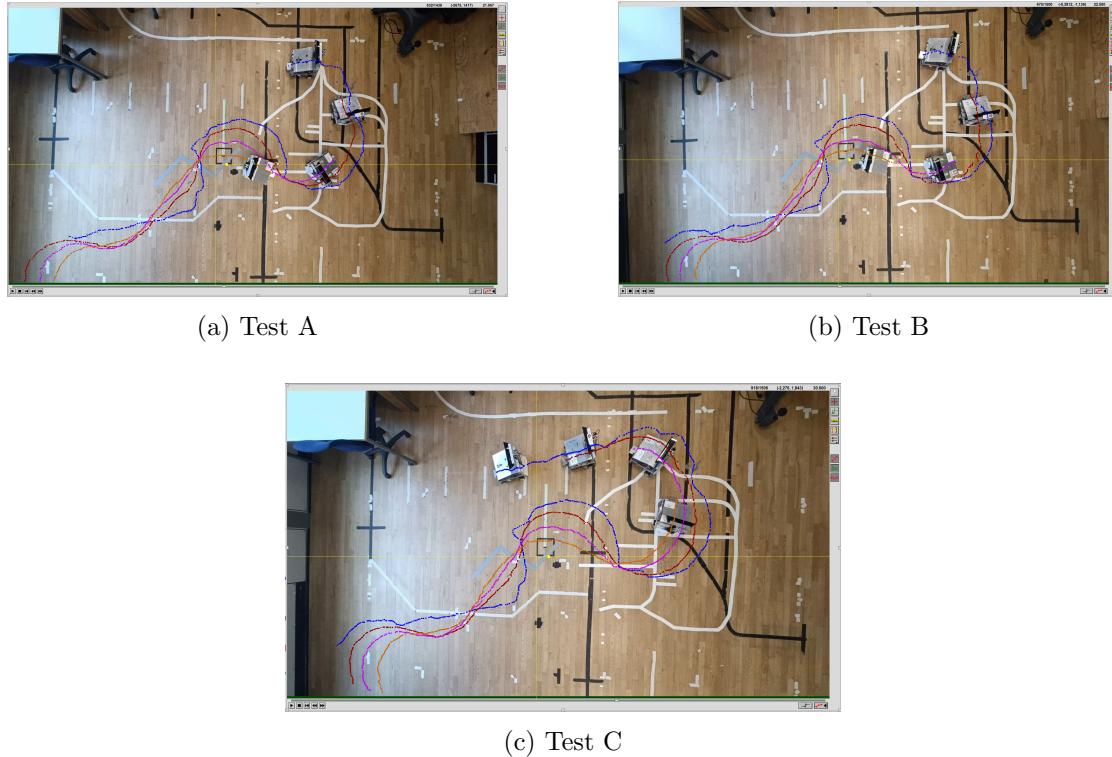
Figur 27: Slut position af robot konvoj.

### 7.1.2 Video Analyse

For hvert forsøg blev der filmet et udsnit af banen ovenfra. Videoen blev indsat i Loggerpro og derfra blev hver robots position manuelt markeret som illustreret i figur 28.

## 7.1 Resultater

## 7 TEST AF IMPLEMENTERING

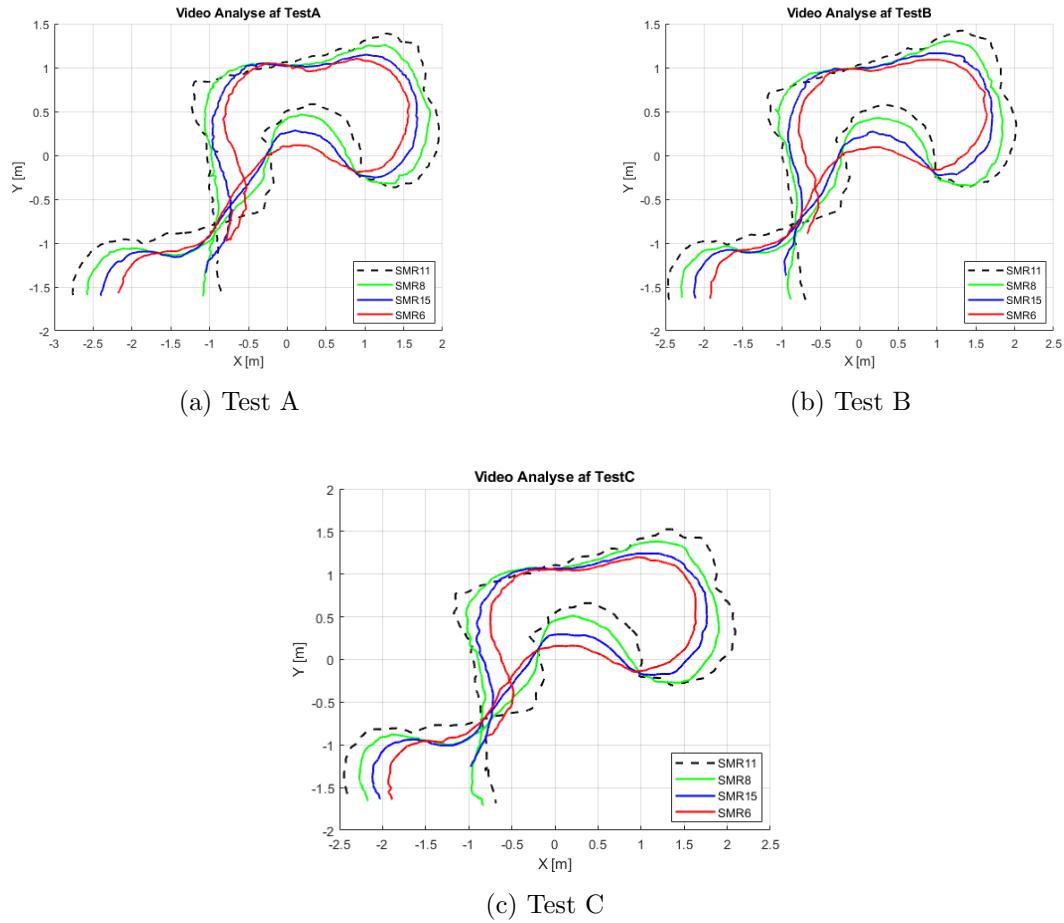


Figur 28: Udtag billede fra videoanalyse i **Loggerpro** for hver test.

Deres positioner blev logget i en .csv-fil med formål at analysere i MATLAB. Som refference er der en linje tape på gulvet, som er 75.5cm lang, og kvadraten i midten af billedet bruges til at indsætte et koordinatsystem (ikke det samme som for odometrien), som er identisk placeret for alle tre forsøg.

## 7.1 Resultater

## 7 TEST AF IMPLEMENTERING



Figur 29: Udtag data fra videoanalyse i Loggerpro for hver test.

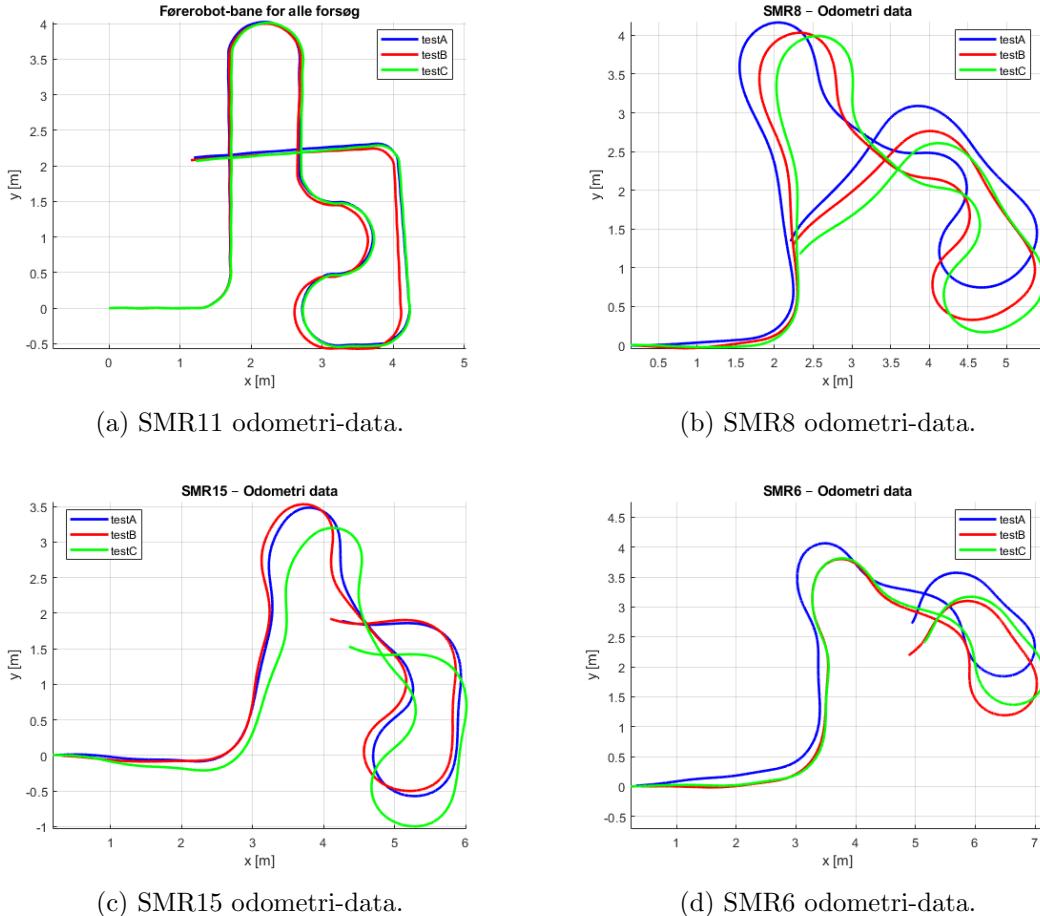
Figur 29 viser henholdsvis **Test A**, **B** og **C**, for hver test er ruten af hver **SMR** robot plottet i et  $x, y$  koordinatsystem.

## 8 Resultatbehandling

Det er interessant at se på forholdene mellem hver følgerobot og førerrobotten, samt deres respektive forankørende. Rækkefølgen for forsøgerne var SMR11 som førerrobot, efterfulgt af SMR8, SMR15, og til sidst SMR6.

### 8.1 Rå odometri-data

Først bemærkes den markante afvigelse mellem følgerobotternes og førerrobottens bane. Årsagen til dette vil blive diskuteret i afsnit 8.2.1. Det er dog kvalitativt tydeligt, at robotterne kører samme rute.

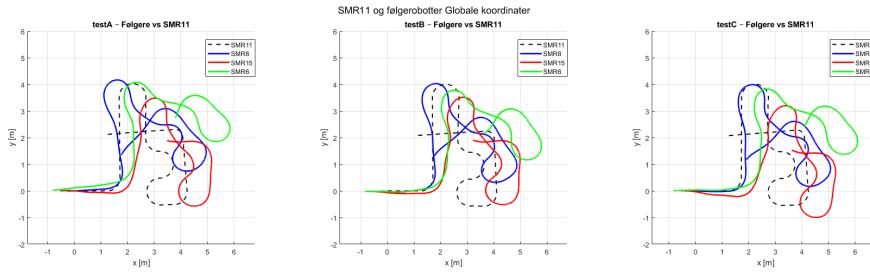


Figur 30: Odometri-data for SMR-robotterne.

Før dataen kan fortolkes, skal alle koordinaterne først transformeres fra hver af de lokale koordinatsystem til ét globalt koordinatsystem. For hver test anvendes startpositionen for førerrobotten (SMR11) som det globale origo, herefter omregnes hvert koordi-

natsæt til dette koordinatsystem via ligning (3.7) (udledt i afsnit 3.3)

$$\begin{bmatrix} x_g \\ y_g \end{bmatrix} = \begin{bmatrix} x_{\text{robot}} \\ y_{\text{robot}} \end{bmatrix} + \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\mathbf{R}(\theta)} \cdot \begin{bmatrix} x_{\text{local}} \\ y_{\text{local}} \end{bmatrix}$$



Figur 31: Odometri-data i globalt koordinatsystem.

For hver følgerobot sammenlignet med førerrobotten, er det af interesse at se, hvor meget hver følgerobot i konvojen afviger fra førerrobottens bane gennem de tre forsøg. Der ses på afvigelsen,

$$E_{\text{path}} = \|\mathbf{x}_r - \mathbf{x}_\ell\|_2 + \|\mathbf{y}_r - \mathbf{y}_\ell\|_2$$

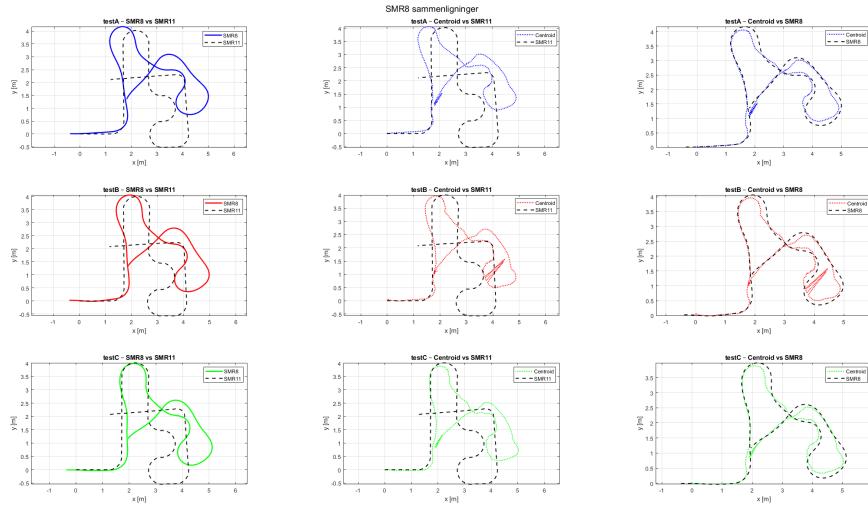
hvor  $\mathbf{x}_r - \mathbf{y}_r$  og  $\mathbf{x}_\ell - \mathbf{y}_\ell$  er henholdsvis følge- og førerrobottens globale koordinater. Der analyseres desuden hvor meget fejlen stiger an på roboternes placering i konvojen. Samme metode anvendes til at sammenligne hver robot med dens forankørende robot, og dermed analyserer, hvor godt hver robot følger den forankørende robot  $E_{\text{robot}} = \|\mathbf{x}_r - \mathbf{x}_\ell\|_2 + \|\mathbf{y}_r - \mathbf{y}_\ell\|_2$ .

## 8.2 Analyse af Odometri-data

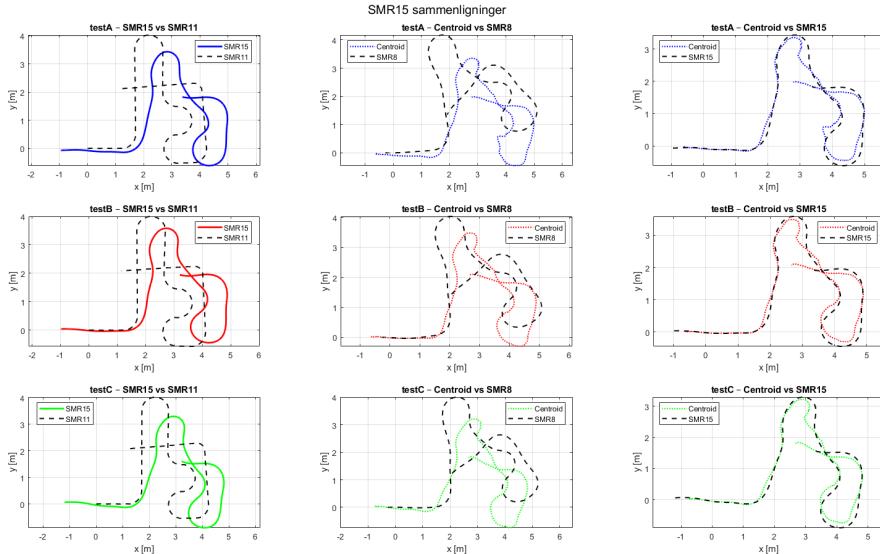
Nedenfor ses dataen fra de forskellige test i det globale koordinatsystem, sorteret efter følgerobotternes placering i konvojen. Rækkerne indikerer testene, henholdsvis A, B, C, og søjlerne indikerer, følgerobotten-SMRxbane sammenlignet med førerrobotten, centroid-sene sammenlignet med den forankørende robots bane, og sidste søjle er følgerobotten-SMRxs rute sammenlignet med centroiden af den forankørende.

## 8.2 Analyse af Odometri-data

## 8 RESULTATBEHANDLING



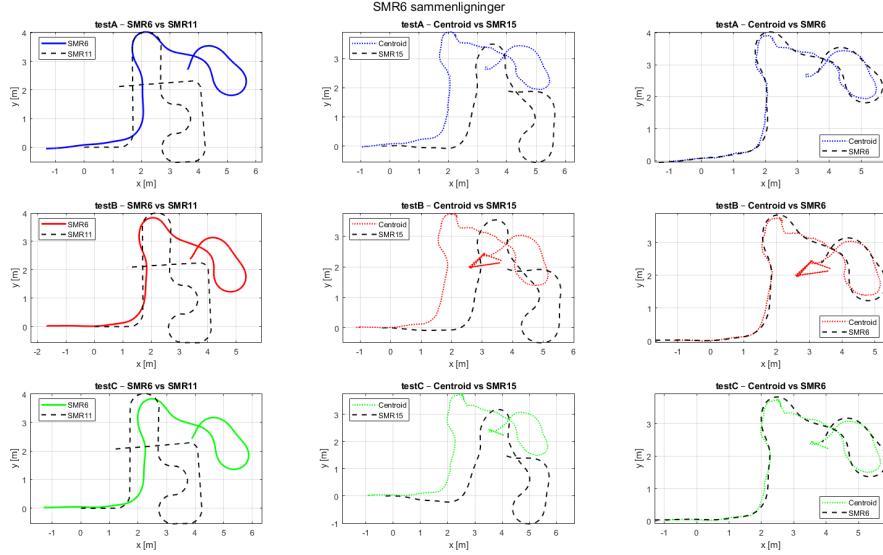
Figur 32: Sammenligning af Data fra SMR 8.



Figur 33: Sammenligning af Data fra SMR 15.

## 8.2 Analyse af Odometri-data

## 8 RESULTATBEHANDLING



Figur 34: Sammenligning af Data fra SMR 6.

Tabel 5: Afvigelse mellem følgeroboterne og førerrobotten  $E^{\text{robot}}$ .

Robot	Test	MeanErr	StdErr	MinErr	MaxErr	MedianErr	Q1	Q3
SMR8	A	1.341	0.594	0.357	2.811	1.257	0.893	1.834
SMR8	B	2.175	0.962	0.107	3.853	2.184	1.364	2.990
SMR8	C	1.856	0.688	0.347	3.181	1.955	1.264	2.402
SMR8	Gennemsnit	1.791	0.748	0.270	3.282	1.799	1.174	2.409
SMR15	A	2.396	1.167	0.227	4.554	2.475	1.544	3.316
SMR15	B	2.875	1.370	0.327	4.779	2.994	1.800	4.111
SMR15	C	2.536	1.269	0.291	4.749	2.557	1.454	3.475
SMR15	Gennemsnit	2.602	1.269	0.282	4.694	2.675	1.599	3.634
SMR6	A	1.566	0.989	0.262	3.485	1.275	0.716	2.187
SMR6	B	1.489	1.041	0.022	3.391	1.403	0.419	2.304
SMR6	C	1.749	0.922	0.311	3.567	1.440	1.021	2.350
SMR6	Gennemsnit	1.601	0.984	0.198	3.481	1.373	0.719	2.280

Tabel 6: Afgelse mellem centroid og forankørende  $E^{\text{centroid}}$ .

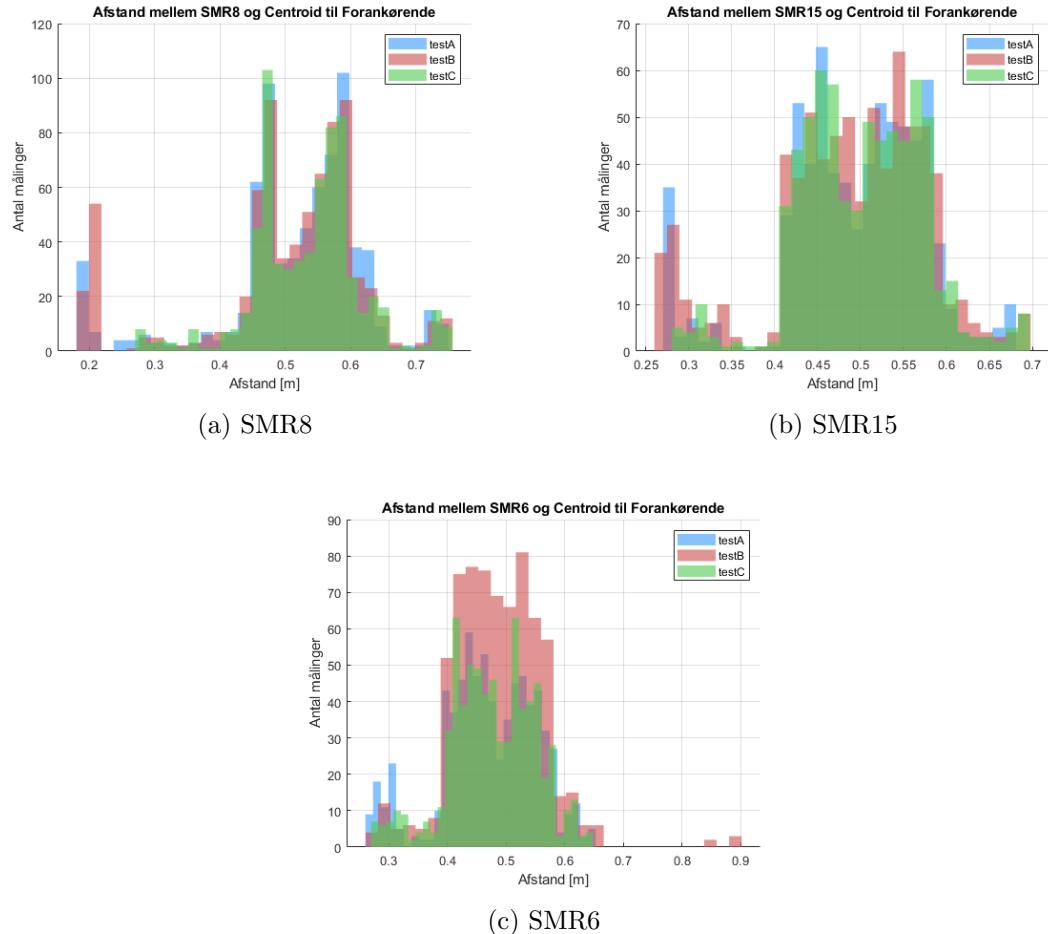
Robot	Test	MeanErr	StdErr	MinErr	MaxErr	MedianErr	Q1	Q3
SMR8	A	1.075	0.505	0	2.201	1.039	0.689	1.448
SMR8	B	1.945	0.833	0	3.473	1.937	1.212	2.750
SMR8	C	1.521	0.654	0	2.871	1.456	1.128	2.015
SMR8	Gennemsnit	1.514	0.664	0	2.848	1.477	1.010	2.071
SMR15	A	1.618	1.189	0	3.571	1.470	0.476	2.878
SMR15	B	1.741	1.079	0	3.281	1.679	0.807	2.893
SMR15	C	1.301	1.037	0	3.039	1.114	0.294	2.426
SMR15	Gennemsnit	1.553	1.102	0	3.297	1.421	0.526	2.732
SMR6	A	1.334	0.993	0	3.083	1.323	0.339	2.124
SMR6	B	1.301	0.943	0	3.149	1.095	0.445	2.061
SMR6	C	1.500	0.939	0	3.196	1.457	0.616	2.207
SMR6	Gennemsnit	1.378	0.958	0	3.143	1.292	0.467	2.131

Tabel 7: Afgelse mellem centroid og egen position  $E^{r \rightarrow c}$ .

Robot	Test	MeanErr	StdErr	MinErr	MaxErr	MedianErr	Q1	Q3
SMR8	A	0.514	0.114	0.184	0.748	0.539	0.447	0.585
SMR8	B	0.502	0.126	0.185	0.755	0.532	0.467	0.584
SMR8	C	0.533	0.146	0.273	0.755	0.548	0.470	0.576
SMR8	Gennemsnit	0.516	0.107	0.214	0.753	0.539	0.471	0.583
SMR15	A	0.493	0.084	0.270	0.680	0.505	0.448	0.555
SMR15	B	0.487	0.092	0.266	0.696	0.495	0.440	0.552
SMR15	C	0.502	0.080	0.283	0.690	0.506	0.453	0.545
SMR15	Gennemsnit	0.494	0.083	0.273	0.690	0.502	0.446	0.554
SMR6	A	0.470	0.083	0.269	0.652	0.469	0.423	0.531
SMR6	B	0.485	0.078	0.274	0.644	0.485	0.437	0.531
SMR6	C	0.477	0.079	0.272	0.732	0.476	0.420	0.531
SMR6	Gennemsnit	0.477	0.078	0.272	0.732	0.476	0.427	0.531

Resultaterne af databehandlingen ses i tabel 5, 6 og 7. Tabellerne viser afvigelserne mellem følgeroboterne og førerrobotten  $E^{\text{robot}}$ , afvigelse mellem centroiden og den forankørende robot  $E^{\text{centroid}}$  og til sidst afvigelse mellem centroid og robottens egen position  $E^{r \rightarrow c}$ .

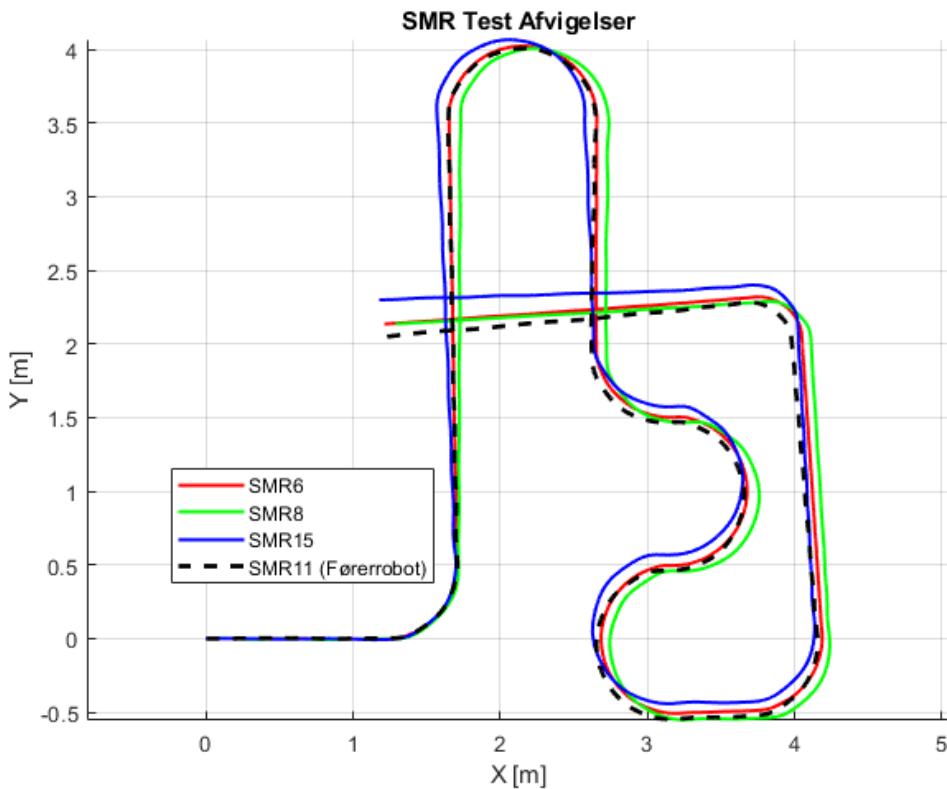
Yderligere ses det på figur 35, at hver SMR-robot har en meget konsistent opfattelse af afstanden til den forankørendes centroid på tværs af forsøgene. Dette tyder på at forsøgene, set fra robottens perspektiv, er identiske og af samme natur.



Figur 35: Afstand mellem hver SMR og dets estimerede centroid til forankørende robot.

### 8.2.1 Analyse af Afgivelse for Ruterne

Der er tydelige fejlkilder på odometriken. For at identificerer hvor fejlkilden præcis ligger, analyseres nogle forskellige scenerier. Først køres hver SMR-robot med førerprogrammet for at se om det er en fejl i odometrikalibreringen. Figur 36 viser tydeligt en ubetydelig afgivelse mellem hver følgerobot (SMR6, SMR8, SMR15) og førerrobotten (SMR11). Resultaterne fra analysen af forsøget ses i tabel 8, og tages dette i betragtning, kan det konkluderes, at afgivelserne ikke skyldes odometriken er fejkalibreret.



Figur 36: Resultat af alle SMR-robotter, der kører førerrobottens program.

Tabel 8: Afvigelse af hver robot og SMR11 (førerprogrammet)

Robot	MeanErr	StdErr	MinErr	MaxErr	MedianErr
SMR6	0.041	0.032	0	0.118	0.038
SMR8	0.093	0.049	0	0.184	0.101
SMR15	0.134	0.091	0	0.322	0.114

Desuden ses det på figur 30b, 30c, og 30d at fejlen må være systematisk, da for hver følgerobot er der en identisk afvigelsestendens på tværs af hver test, og da hver test er uafhængige af hinanden, kan det ekskluderes, at afvigelserne skyldes akkumuleret fejl fra en test til en anden. Det er muligt, at afvigelserne i odometrien skyldes, at programmet for følgeroboterne sender kommandoer med en relativ høj frekvens (10/s), med små variationer i hastighedskommandoerne. Dette kan potentielt forklare hvordan fejlene systematisk bliver akkumuleret, da odometrien udregnes som

$$\begin{aligned}
 x(i) &= x(i-1) + \Delta U(i) \cos(\theta(i)) \\
 y(i) &= y(i-1) + \Delta U(i) \sin(\theta(i)) \\
 \theta(i) &= \theta(i-1) + \Delta \theta(i) \\
 \Delta U(i) &= \frac{\Delta U_R(i) + \Delta U_L(i)}{2} \\
 \Delta \theta(i) &= \frac{\Delta U_R(i) - \Delta U_L(i)}{b}
 \end{aligned}$$

hvor,  $\Delta U_{L/R}$  er ændringen i position udregnet ud fra en 'quaddencoder' [Stu25]. På grund af de mange hastighedskommandoer er det muligt, at robotterne skrider en lille smule, og dermed akkumulerer afvigelserne systematisk. Ydermere, kan heller ikke ekskluderes, om afvigelserne er afhængige af robotternes placering i konvojen, eller om det er robot-specifict. Vurderingen af dette er dog uddover rapportens rammer.

Af disse grunde er det værd at påpege, at dataen fra LiDAR-sensoren i tabel 7 stadig er relevant, da dataen fra LiDAR-sensoren udregner datapunkternes koordinater ud fra robottens odometri, og dermed har de samme bias. Opsummeret er der høj bias, men lav varians for forsøgene.

### 8.3 Video Analyse

For at få en uafhængig analyse af robotterne, blev et udsnit af hvert forsøg filmet ovenfra. Videoerne blev så analyseret i **Loggerpro**.

#### 8.3.1 Korrektion af Filmvinkel

Der er en fejlkilde i form af filme-vinklen i forhold til gulvet. Der er et lille kvadrat omkring midten af video, hvor i virkeligheden er hver vinkel  $90^\circ$ , men for videoerne er disse vinkler forskudt. Et eksempel kan ses i figur 37. Alle vinklerne blev udledt ved hjælp af **GeoGebra**. Resultaterne kan ses i tabel 9.

Tabel 9: Indvendige hjørnevinkler målt i hvert testområde.

Test	Øvre venstre [°]	Øvre højre [°]	Nedre højre [°]	Nedre venstre [°]
A	89.75	90.77	89.35	90.12
B	89.32	91.62	88.71	90.35
C	89.30	93.87	89.25	87.58



Figur 37: Vinkel forvridning for video analyse af Test C.

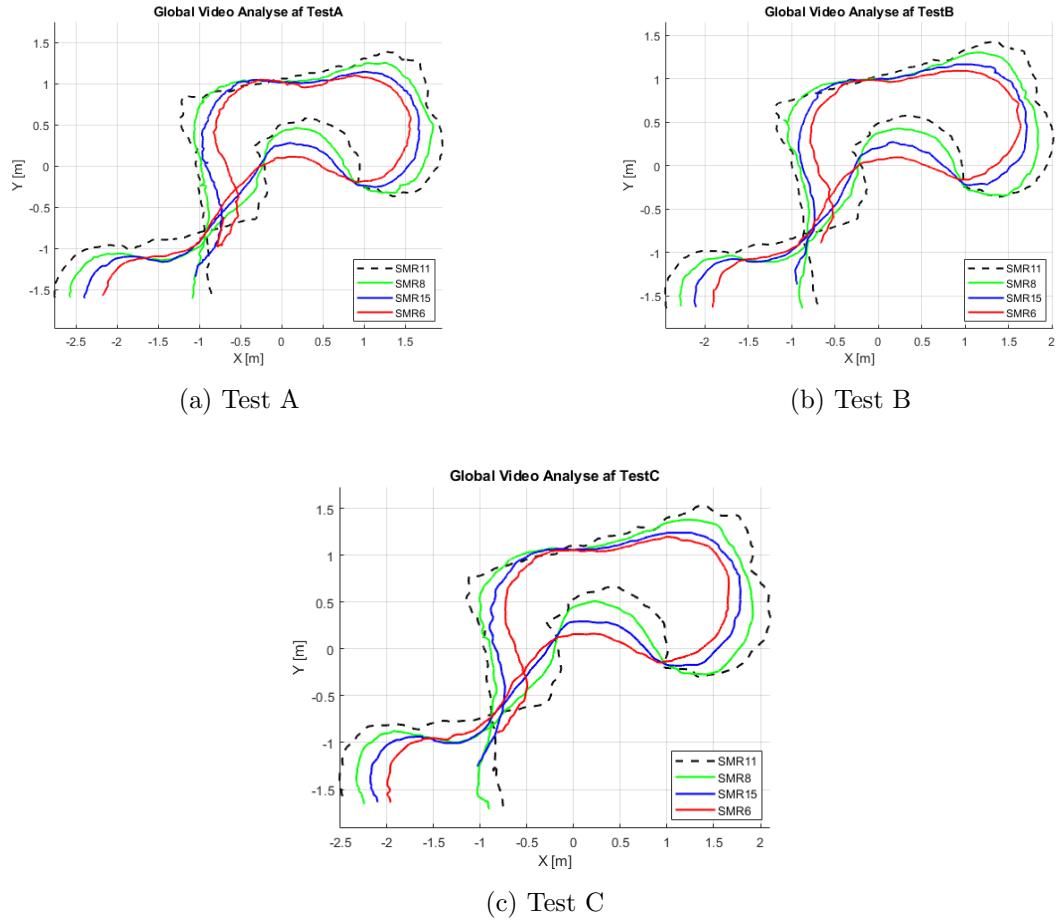
Det er muligt at lave en tilnærmet rettelse af denne forvridning (skewness) i vinklerne, og dermed få ét koordinatsystem, hvorfra hver test kan sammenlignes. For hver video er koordinatsystemet placeret i det nedre venstre hjørne af kvadraten. Ved at definerer en basisvektor i det forvrede koordinatsystem, udtrykt som dens reelle koordinater, kan en basis udledes:

$$\begin{aligned}\mathbf{e}_x &= \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \\ \mathbf{e}_y &= \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}, \\ \mathbf{S} &= \begin{bmatrix} 1 & \cos(\theta) \\ 0 & \sin(\theta) \end{bmatrix}\end{aligned}$$

Herfra kan koordinatskiftet så udregnes således:

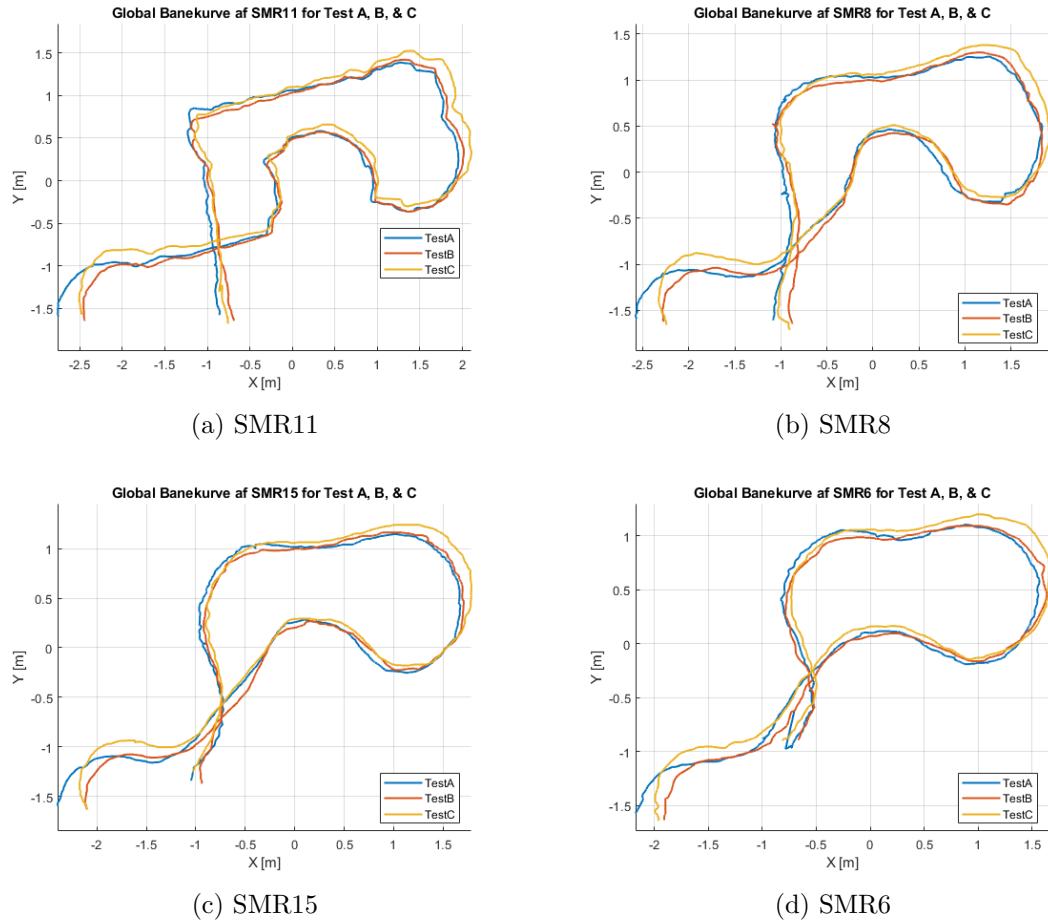
$$\mathbf{p}_{\text{global}} = \mathbf{S} \cdot \mathbf{p}_{\text{skewed}} = \begin{bmatrix} 1 & \cos(\theta) \\ 0 & \sin(\theta) \end{bmatrix} \begin{bmatrix} x_s \\ y_s \end{bmatrix}$$

Ved at anvende denne transformation for hver vinkel opnås et tilnærmet globalt koordinatsystem for alle testene.



Figur 38: Videoanalyse i Loggerpro for hver test i det globale koordinatsystem.

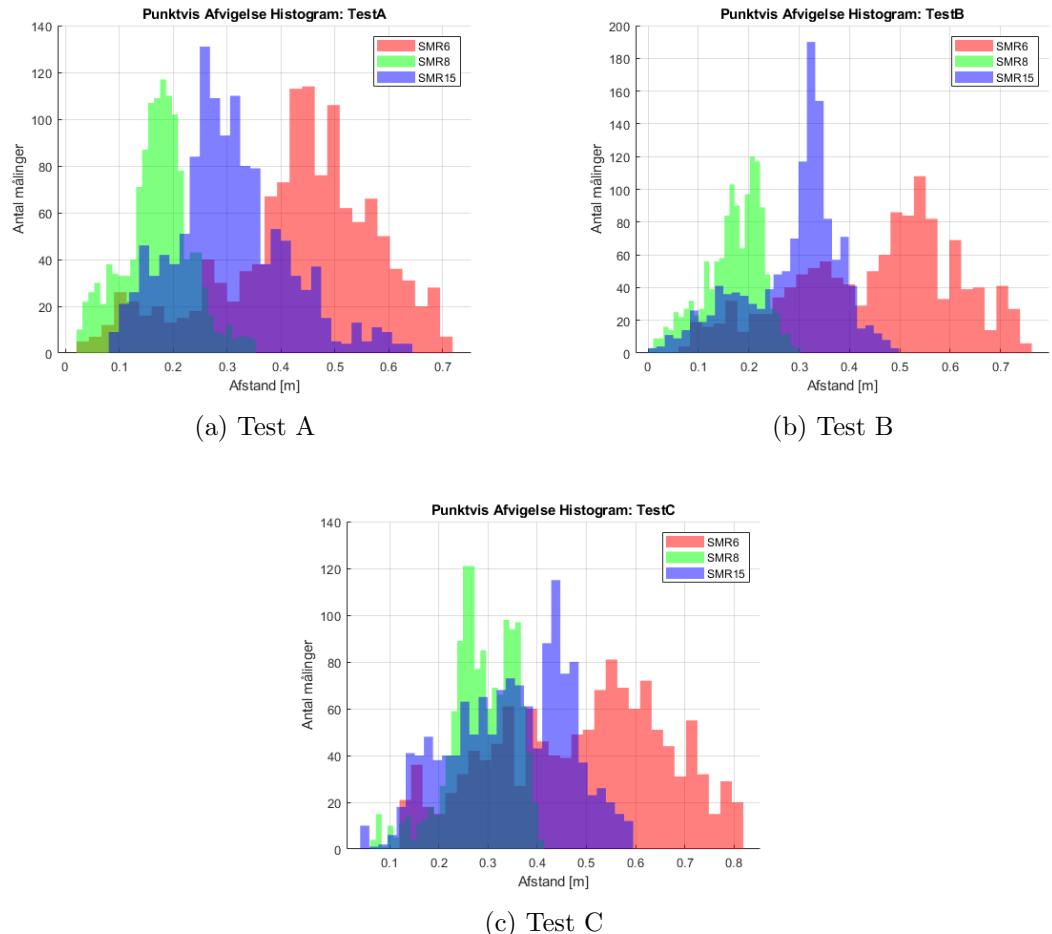
En interessant sidebemærkning er, selv om der var en fejkilde i form af filmvinklen, så formår hver robot at ligge meget ens fra den ene test til den anden. **Test C** afviger mest fra de to andre, da det er den med største forvridning i vinklen. Dette betyder altså at alle testene var i realiteten meget identiske, hvilket ses i figur 39.



Figur 39: Sammenligning af hver robots banekurve for hver test.

### 8.3.2 Afvigelse medhensyn til Førerrobotten

Samme analyse som der blev anvendt for odometriken, bliver anvendt på dette data. Der ses på den totale afstand mellem hver følgerobot og førerrobotten for hver test. Resultaterne af dette ses i tabel 10.



Figur 40: Histogrammer over punktvis afstandsfejl mellem følgerobotter og førerrobotten i de tre tests.

Tabel 10: Afstandsafvigelse mellem følgeroboternes til førerrobottens position ud fra videodata.

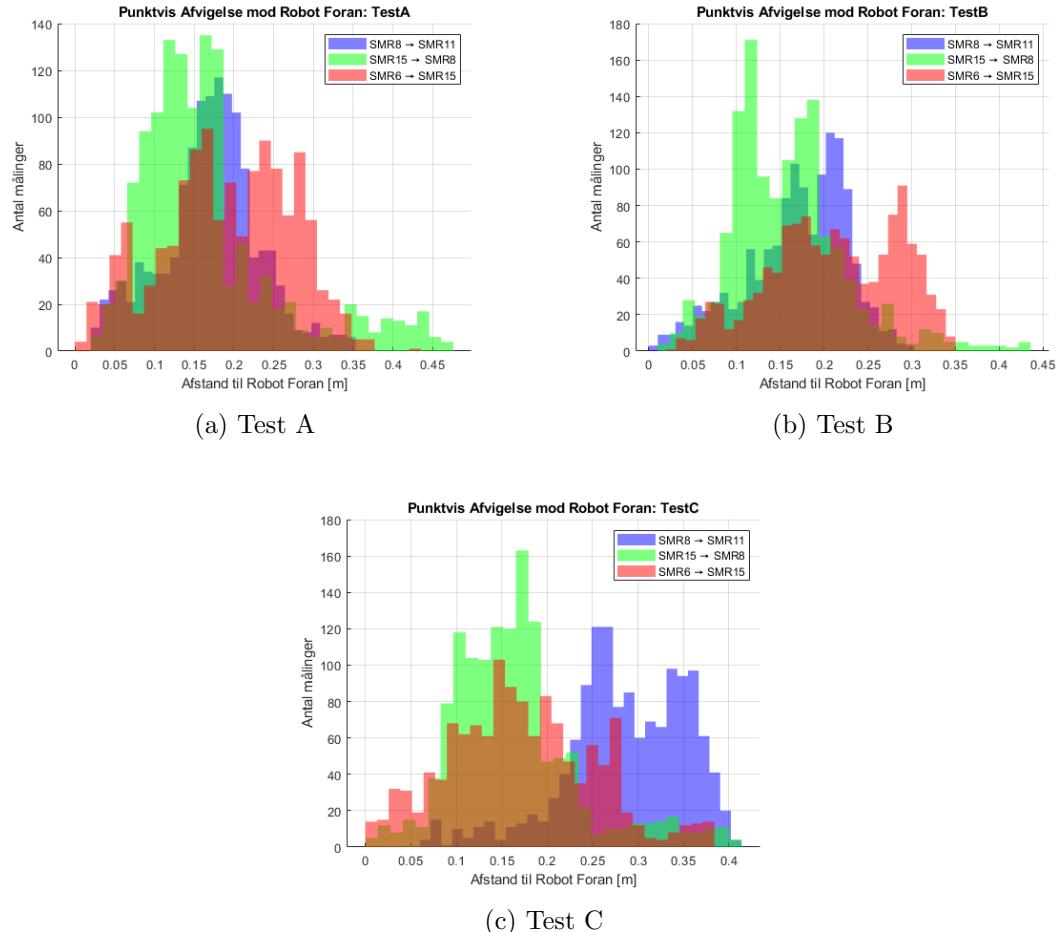
Test	Robot	$\mu$ [m]	$\sigma$ [m]	Min. fejl [m]	Maks. fejl [m]	Median [m]	Q1 [m]	Q3 [m]
TestA	SMR6	0.614	0.244	0.036	1.057	0.592	0.443	0.798
TestA	SMR8	0.192	0.081	0.004	0.340	0.200	0.135	0.261
TestA	SMR15	0.368	0.157	0.010	0.645	0.386	0.259	0.507
TestB	SMR6	0.674	0.257	0.112	1.100	0.680	0.482	0.896
TestB	SMR8	0.192	0.070	0.025	0.328	0.192	0.142	0.255
TestB	SMR15	0.336	0.136	0.015	0.567	0.343	0.243	0.457
TestC	SMR6	0.710	0.260	0.167	1.155	0.720	0.503	0.942
TestC	SMR8	0.260	0.063	0.047	0.375	0.263	0.230	0.307
TestC	SMR15	0.443	0.155	0.143	0.742	0.428	0.312	0.595

### 8.3.3 Afvigelse medhensyn til Forankørende

Igen på sammevis behandles dataen for distanceafvigelsen mellem hver robot og den forankørende. Som tabel 11 og figurerne 41 viser, så er gennemsnitsafvigelsen af distancen mellem testene forskellige. For Test A ligger SMR8 og SMR15 meget ens i forhold til deres robototter foran, mens SMR6 har den største varians og største gennemsnitsafvigelse.

### 8.3 Video Analyse

## 8 RESULTATBEHANDLING



Figur 41: Afstand mellem hver robot og den forankørende robot under de tre tests.

Tabel 11: Afstand mellem hver følgerobot og den forankørende robot målt ud fra video-data.

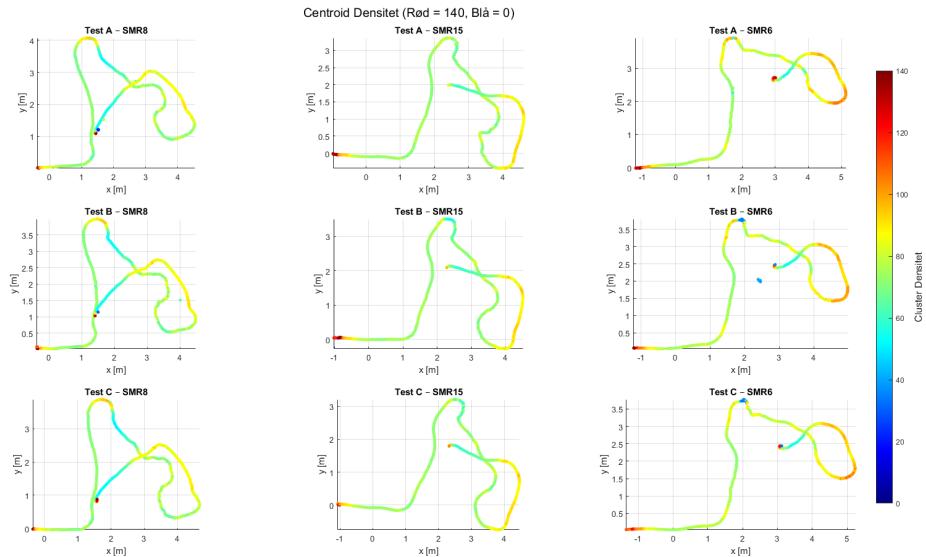
Test	Robot	$\mu$ [m]	$\sigma$ [m]	Min. fejl [m]	Maks. fejl [m]	Median [m]	Q1 [m]	Q3 [m]
TestA	SMR8	0.192	0.081	0.004	0.340	0.200	0.135	0.261
TestA	SMR15	0.197	0.085	0.008	0.392	0.195	0.137	0.256
TestA	SMR6	0.282	0.115	0.005	0.885	0.299	0.214	0.374
TestB	SMR8	0.192	0.070	0.025	0.328	0.192	0.142	0.255
TestB	SMR15	0.172	0.070	0.021	0.306	0.182	0.109	0.226
TestB	SMR6	0.383	0.148	0.062	0.713	0.425	0.249	0.489
TestC	SMR8	0.260	0.063	0.047	0.375	0.263	0.230	0.307
TestC	SMR15	0.223	0.097	0.019	0.472	0.225	0.145	0.305
TestC	SMR6	0.290	0.123	0.007	0.489	0.319	0.184	0.393

## 9 Diskussion

I dette afsnit diskutes betydning af resultbehandlingen for de forskellige forsøg, samt bliver datasættenes resultater sat i betragtning i forhold til hvad formålet med rapporten er.

### 9.1 Cluster-densitet

Idet centroid-algoritmen 2 filtrerer LiDAR-dataen efter cluster-densitet, altså hvor mange punkter, der er i et givet cluster, er det vigtigt at analysere om det udvalgte interval var passende. Udfra figur 42, som viser hver følgerobots centroid-densitet i form af et heatmap, er det åbenlyst, at intervallet [20; 140] var passende.



Figur 42: Antal cluster vist som et heatmap for hver følgerobot for hver test.

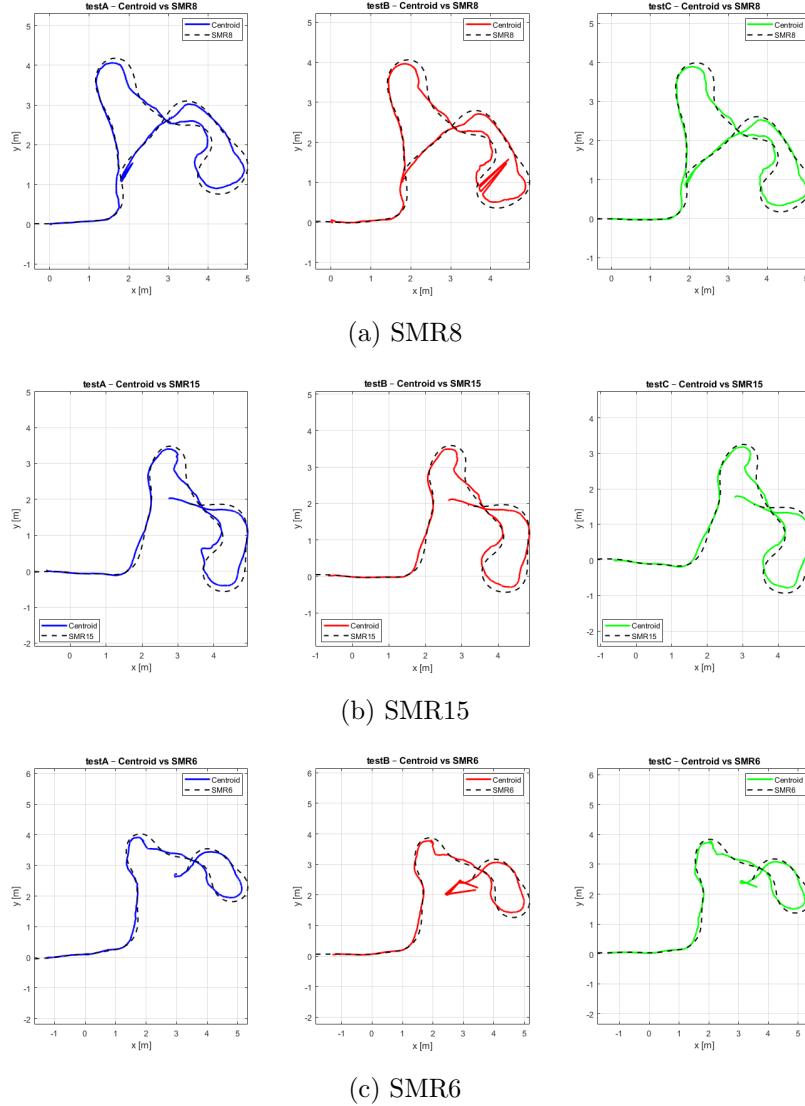
Farverne er ens gennem store dele af ruten, med undtagelse af starten, hvor der er få punkter med meget høj densitet. Det eneste bemærkelsesværdig punkt er **Test B** for **SMR6**, hvor der ses et hul efterfulgt af punkter med en middel til lav densitet. Dette tyder på, at robotten mistede den forankørende robot, men fangende den igen. Hvilket er værd at påpege, da det betyder, at algoritmen 2 fungerer efterhensigt på dette aspekt.

### 9.2 Centroid (Odometri)

Centroiden formår tydeligt at følge den forankørende robot med små afvigelser. Alle **SMR**-robotterne havde en næsten identisk gennemsnitsafstand-afvigelse på omkring  $\mu \approx 0.5m$ , som ses i tabel 7. Ligeledes ses det, at hoveddelen af afvigelserne er akkumuleret i svingene, hvilket er udtrykt i figur 43.

## 9.2 Centroid (Odometri)

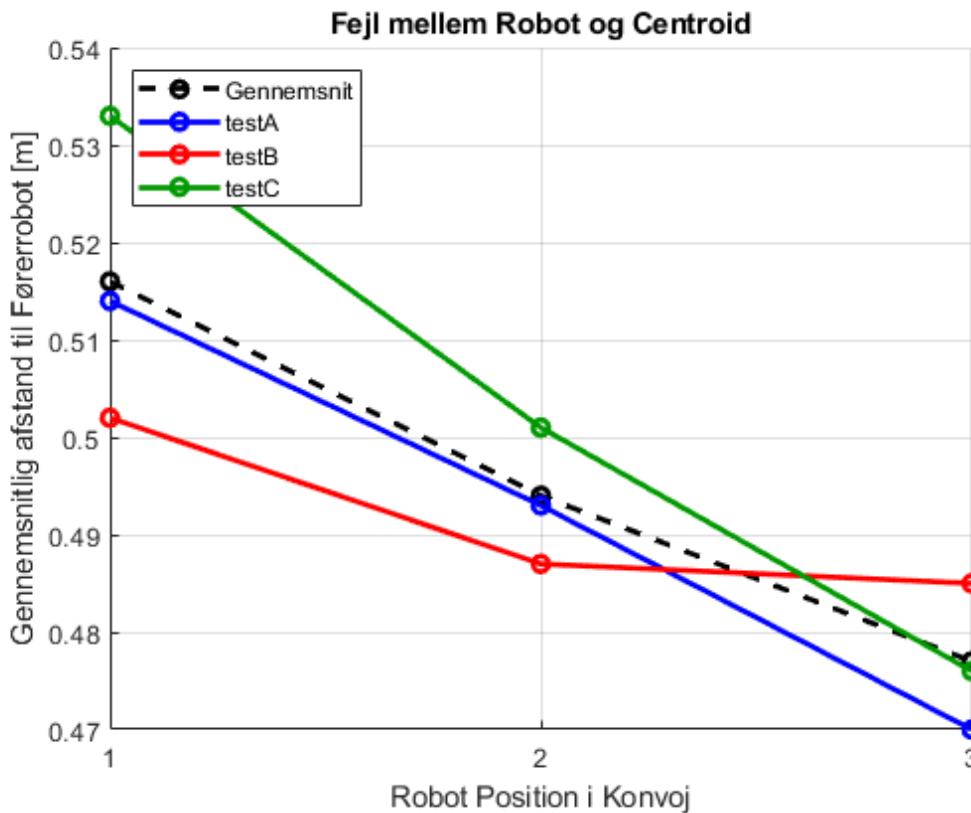
## 9 DISKUSSION



Figur 43: Centroidpositioner for SMR-robotterne.

Det bemærkes, at centroiden mere hyppigt ligger indenfor grænsen af den forankørende, hvilket vil sige, at robotten ser den forankørende robot som værende tætter på end den egenligt er i virkeligheden. Det betyder at forskydningen af centroiden ikke er nok, og enten skal den forskydes mere, eller en anden metode skal bruges, hvis en bedre centroid-tracking skal opnås. Der er også tilfælde, som ses på figur 43a for Test B, hvor der ses et hak i ruten af centroiden. Dette er et punkt hvor centroid-detecteringen ikke fungerede korrekt, hvilket resulterede i et spring i centroidens lokation. I logfilen `full_log8.csv` for Test B, ved indeks 452 ses det, at centroiden er fundet, og har koordinaterne  $x_{local}, y_{local} = (4.088, 0.878)$  og i næste indeks 453 er der opfangeet en fejl (`ClusterSize` viser  $-1$ , altså en fejl), da centroiden springer til  $x_{local}, y_{local} =$

(4.847, 1.552) og næste indeks ligger igen på banen. Springet kan skyldes forskellige årsager, f.eks. en solstråle, som reflekterer og danner interferens med LiDAR-sensoren, og dermed giver en fejlmåling osv. En enkel outlier som denne, er dog alene ikke nok til at forklare gennemsnitsfejlen. Det kan heller ikke skyldes sensorens præcision på  $\pm 3\%$ .



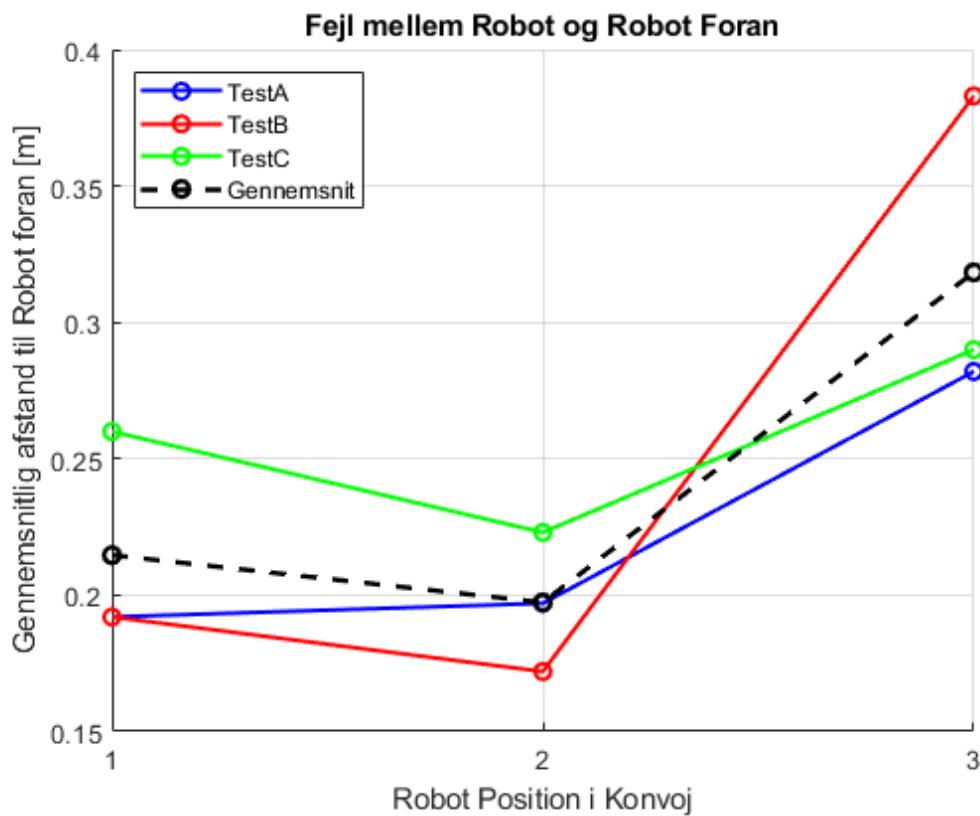
Figur 44: Fejl mellem robot og centroid i forhold til position i konvojen.

Yderligere skal det påpeges, at gennemsnitsfejlen for centroidene har en nedadgående tendens, når positionen i konvojen tages i betragtning. Dette skyldes, at robotterne skærer af hjørner, og dermed kører i mere direkte linjer. SMR8 ser på førerrobotten og dermed følger en rute, som ikke er ligeså udglattet, som eksempelvis SMR6 (den bagerste i konvojen) følger. SMR6 prøver at følge en robot SMR15, som skærer af hjørner og dermed gør ruten mere glat. Altså, ses der en nedad gående tendens, da hver robot opfatter den forankørende, som at køre mere glat, end den reelt set gør.

Da robotterne kører efter centroiden, og der er en gennemsnitsfejl på  $0.5m$ , i følge cluster dataen, er det interessant at sammenligne med data fra videoanalysen.

### 9.3 Forankørende (Videodata)

Som tabel 11 og figurerne 41 viser, så er gennemsnitsafvigelsen af distancen mellem testene forskellige fra dataen fra centroid-analysen i forrige afsnit. Figur 45 viser, at for konvojposition 1 og 2 falder fejlen, men for position 3, stiger fejlen. Det tyder på, at i virkeligheden har robotten i position 3 (SMR6) en tydelig tendens til at følge den forankørende robot dårligere. Det er dog, som nævnt før, ikke muligt at afgører om det skyldes den specifikke robot SMR6, eller om det er afhængigt af konvoj positionen, ud fra testene.

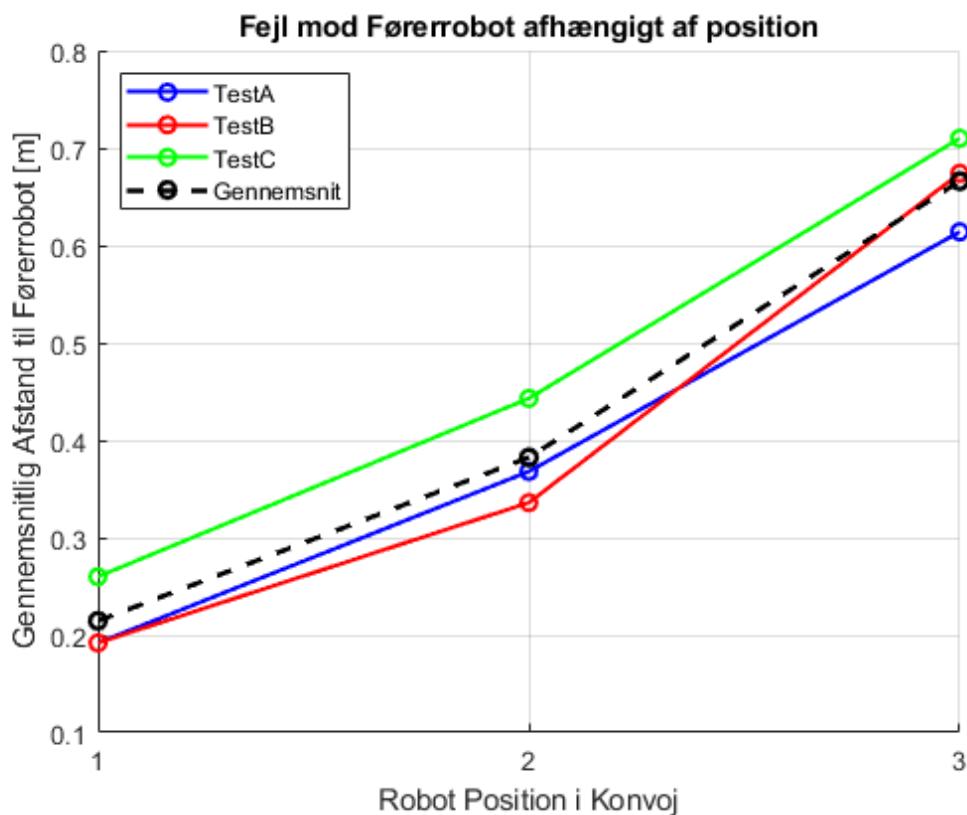


Figur 45: Gennemsnitsafstandsfejlen som funktion af konvojposition.

Det er dog åbenlyst, at i virkeligheden (eller mere specifikt, set ovenfra og analyseret ud fra en video) er fejlen mindre, end hvad odometri- og centroid-dataen tyder på. Det ligner endda, at distanceafvigelsen, vurderet ud fra videoanalysen for videoanalysen, er tilnærmelsesvis (for de første to positioner) halvdelen af hvad odometriken viser. Dette medfører altså at den potentielle outlier, som er nævnt i forrige afsnit, ikke havde en indflydelse på resulterne.

#### 9.4 Førerrobot (Videodata)

Som forventet stiger gennemsnitsafvigelsen fra den rute som førerrobotten kørte som funktion af position i konvojen. Dette skyldes, som nævnt tidligere, at robotterne skærer hjørner af, og dermed akkumulerer fejlen som funktion af position, hvilket tydeligt ses i figur 46. Som nævnt i afsnit 8.3.1, så er det forudsigligt at Test C vil have den største fejl, hvilket sandsynligvis skyldes filmvinklens forvridning var største for dette forsøg. Dette er forklaringen, hvorfor den ligger konsistent højere end de to andre.



Figur 46: Gennemsnitsfejl i forhold til førerrobotten.

#### 9.5 Kvalitativ Analyse

I virkeligheden ser konvojen meget god ud, og det er svært at bemærke position-afvigelserne i konvojen. De eneste steder, hvor det er synligt, er i starten, hvor robotterne skal kører gennem døråbningen. Her ses det, at robotterne kommer tætter og tætter op ad venstre side. I slutningen af ruten, hvor førerrobotten øger hastigheden fra  $0.2\text{m/s}$  til  $0.4\text{m/s}$  observeres det, at de bagerste robotter ikke har nået at tage det sidste sving inden de accelererer, og dermed overkurregerer mod venstre (set fra deres perspektiv) inden de retter op for så at ende bag hinanden. Udover disse to scenerier er det svært at observere disse

afvigelser, hvis der ikke sammenlignes med et referencepunkt, såsom tapen på gulvet. Det er også ret evident, at ved laverer hastigheder ( $0.2m/s$ ), og på lige strækninger, så følger robotterne hinanden i en konvoj meget bedre hvilket er ret åbenlyst og forventet.

## 9.6 Forbedringsforslag og Videreudvikling

Selvfølgelig, som med alle projekter, er der altid plads til forbedring og videreudvikling, men følgende er af særlig interesse:

- Det kunne være meget interessant at isolere den præcise årsag til, hvorfor odometri-daten har den afvigelse som den har, eller endda helt løse problemet, da dette ville give et meget mere præcist indblik i, hvordan hver robot vurderer hvor godt den kører efter den forankørende og førerrobotten, samt give muligheden for at sammenligne odometri-data med video-data, da disse burde illustrerer tilnærmedesvist de samme kurver.
- Det kunne være interessant at udfører alle forsøgene med hver robot i hver konvojposition (24 forsøg i alt) og derfra udlede, om gennemsnitsfejlen alene skyldes roboternes position i konvojen, eller om robotterne har en karakteristisk afvigelse. I bagklogskabens klare lys er dette en relativ åbenlys undersøgelse.
- I stedet for at udlede en regulator, først uden en estimeret sensor i feedback-loopet, og dernæst med, kunne en masse tid have været sparet ved at starte ved at tage højde for det tidligere i forløbet, og dermed opnå en optimal steprespons for hastighedsregulatoren.
- Det kunne ligeledes være interessant at test præcist hvor grænsen er for antallet af følgerobotterne. Altså undersøge om gennemsnitsfejlen er aftagende og tillader et ubegrænset antal robotter i konvojen, eller om der er en øvregrense for antallet mm.
- Da robotterne er udstyret med et Xbox-360 Kinect kamera, kunne det være af interesse at anvende det til objektgenkendelse, og eventuelt bruge det til at sammenligne, hvad det ser med LiDAR-sensorens data, for så yderligere at sortere cluster-dataen.
- En potentiel videreudvikling, eller som et andet bachelorprojekt, kunne være at implementere 'Model Predictive Control' [BABF22], hvor der forudsiges hvad den næste forventede respons burde være, og tage det i betragtning i kontrolstrategien.

## 10 Konklusion

Det implementerede LiDAR-baserede konvojsystem gør det muligt for en gruppe af DTU's SMR-robotter at følge en førerrobot, der kører en planlagt rute. Ved hjælp af PI-Lead regulatoren for hulhastighederne samt realtidsdata fra LiDAR-sensoren er robotterne i stand til at fastholde en fast konvojformation, og afslutter deres rute i en linje bag førerrobotten. Systemet fungerer tilfredsstillende i de fleste scenarier, dog med nogle begrænsninger, som resulterede i små afvigelser og hjørneafskæring. Med udgangspunkt i analysen udført i denne rapport, er det muligt at udarbejde disse begrænsninger og dermed opnå et endnu bedre og tilfredsstillende konvojsystem med reduceret hjørneafskæring og afvigelser.

## Litteratur

- [AR04] N. A. Andersen and O. Ravn. SMR-CL: A real-time control language for mobile robots. In *Proceedings of the 2004 CIGR International Conference*, Beijing, China, 2004. [Online]. Available: <https://orbit.dtu.dk/en/publications/smr-cl-a-real-time-control-language-for-mobile-robots>. Accessed: Feb. 11, 2024.
- [ASN22] J. C. Andersen, I. Santos, and H. H. Niemann. *Feedback Control Techniques: A PID Design Handbook*. Polyteknisk Forlag, Copenhagen, Denmark, 1 edition, 2022.
- [BABF22] Aleksandr P. Bondarchuk, Pavel V. Abramov, Svetlana M. Bogdanova, and Denis M. Filatov. Mobile robot model predictive control model. In *Proceedings of the 2022 25th International Conference on Soft Computing and Measurements (SCM)*, pages 53–55, St. Petersburg, Russia, 2022. Institute of Electrical and Electronics Engineers Inc.
- [Hok25] Hokuyo Automatic Co., Ltd. Ust-10lx – scanning laser range finder. <https://www.hokuyo-aut.jp/search/single.php?serial=166>, 2025. Accessed: May 30, 2025.
- [Mat25] MathWorks. compare (system identification toolbox). <https://se.mathworks.com/help/ident/ref/compare.html>, 2025. Accessed: May 13, 2025.
- [Reg25] Regneregler.dk. Cirkel – korde. <https://regneregler.dk/cirkel-korde/>, 2025. Accessed: May 13, 2025.
- [Stu25] Studocu. 31385 exercise 2 – odometry. <https://www.studocu.com/danmarks-tekniske-universitet/autonome-robotssystemer/31385-exercise-2-odometry/3991248>, 2025. Accessed: June 5, 2025.
- [Web25] Webmatematik. Krumning. <https://www.webmatematik.dk/lektioner/saerligt-for-htx/differentialregning/krumning>, 2025. Accessed: May 13, 2025.
- [Wik25] Wikipedia contributors. Cluster analysis. [https://en.wikipedia.org/wiki/Cluster\\_analysis](https://en.wikipedia.org/wiki/Cluster_analysis), 2025. Accessed: Feb. 25, 2025.

## Figurer

1	Billede af de fire SMR robotter, som indgår i konvojen, i deres startposition.	4
2	Skitse af konvojsystem med relevante variabler for opstilling af kinematisk model.	6
3	To datasæt, der viser stepresponsen af SMR-Robot ved en hastighedskommando på $0.5 \frac{m}{s}$ .	11
4	Overføringsfunktion vs valideringsdata	12
5	Et reguleringsssystem med feedback og relevante signalnavne.	13
6	Steprespons plots af P-controller med to forskellige gain-værdier.	14
7	Steprespons for PI-Regulator.	15
8	Step af respons af teoretisk optimal PI-Lead-Regulator.	16
9	Diskret PI-Lead af den optimale kontinuerlige PI-Lead.	17
10	Rå LiDAR-sensor uden fortolkning.	19
11	Filtret data baseret på distance.	20
12	Bedste Cluster (grøn) valgt baseret på algoritme.	21
13	Centroid udregnet baseret på samme data fra figur 12.	23
14	Oscillerende hjulhastigheder af følgerobot, der følger en robot, som kører $0.2m/s$ .	25
15	Modellering af systemet som et blokdiagram i Simulink.	25
16	Modellering af oscillerende hjulhastigheder.	26
17	Forbedret hjulhastigheder af følgerobot, der følger en robot, som kører $0.2m/s$ .	28
18	Modellering af den endelige implementering i samme scenarie.	29
19	Sammenligning af den estimeret model af robotten, den første implementation, og den endelige implementation	30
20	Plot af den gennemsnitlige fejl, det udvalgte data, og reference distancen.	31
21	Histogram af $e[0]$ .	32
22	Illustration af centroid forskydning.	33
23	Situationen hvor $M' = P_d = P_{Centroid}$ , robotterne kører på samme linje.	34
24	Flowchart af C-programmet, der styrer følgeroboterne.	36
25	Ruten som førerrobotten kører, hvor farverne indikerer hastighederne (rød = $0.4m/s$ , grøn = $0.3m/s$ , blå = $0.2m/s$ ), hvor pilene indikerer kørselsretningen.	39
26	Start position af konvojen.	40
27	Slut position af robot konvoj.	42
28	Udtag billede fra videoanalyse i Loggerpro for hver test.	43
29	Udtag data fra videoanalyse i Loggerpro for hver test.	44
30	Odometri-data for SMR-robotterne.	45
31	Odometri-data i globalt koordinatsystem.	46
32	Sammenligning af Data fra SMR 8.	47
33	Sammenligning af Data fra SMR 15.	47
34	Sammenligning af Data fra SMR 6.	48

35	Afstand mellem hver SMR og dets estimerede centroid til forankørende robot. . . . .	50
36	Resultat af alle SMR-robotter, der kører førerrobottens program. . . . .	51
37	Vinkel forvridning for video analyse af Test C. . . . .	53
38	Videoanalyse i Loggerpro for hver test i det globale koordinatsystem. . . . .	54
39	Sammenligning af hver robots banekurve for hver test. . . . .	55
40	Histogrammer over punktvis afstandsfejl mellem følgeroboter og førerrobotten i de tre tests. . . . .	56
41	Afstand mellem hver robot og den forankørende robot under de tre tests. . . . .	58
42	Antal cluster vist som et heatmap for hver følgerobot for hver test. . . . .	59
43	Centroidpositioner for SMR-roboterne. . . . .	60
44	Fejl mellem robot og centroid i forhold til position i konvojen. . . . .	61
45	Gennemsnitsafstandsfejlen som funktion af konvojposition. . . . .	62
46	Gennemsnitsfejl i forhold til førerrobotten. . . . .	63

## Tabeller

1	Indhold i <code>control_logX.csv</code> ( $X = 6, 8, 15$ ) . . . . .	41
2	Indhold i <code>full_logX.csv</code> ( $X = 6, 8, 15$ ) . . . . .	41
3	Indhold i <code>log</code> (førerrobot) . . . . .	41
4	Data uddrag fra <code>full_log15.csv</code> (Test A) . . . . .	42
5	Afvigelse mellem følgeroboterne og førerrobotten $E^{\text{robot}}$ . . . . .	48
6	Afvigelse mellem centroid og forankørende $E^{\text{centroid}}$ . . . . .	49
7	Afvigelse mellem centroid og egen position $E^{\text{r} \rightarrow \text{c}}$ . . . . .	49
8	Afvigelse af hver robot og SMR11 (førerprogrammet) . . . . .	51
9	Indvendige hjørnevinkler målt i hvert testområde. . . . .	52
10	Afstandsafvigelse mellem følgeroboternes til førerrobottens position ud fra videodata. . . . .	57
11	Afstand mellem hver følgerobot og den forankørende robot målt ud fra videodata. . . . .	58

## List of Algorithms

1	Simpel Centroid-Detektion . . . . .	23
2	Implementeret Centroid-Detektion . . . . .	37

## 11 Bilag

Alle bilag er vedlagt digital ved afleveringen, men kan også findes her [GitHub](#).